



Technical Report

# **Deploying IBM Spectrum Scale with NetApp E-Series Storage**

## **Installation and Validation**

Chris Seirer, NetApp  
March 2021 | TR-4859

### **Abstract**

This technical report describes the process of deploying a full parallel file system solution based on IBM's Spectrum Scale™ software stack. This document is designed to provide details on how to install Spectrum Scale, validate the infrastructure, and manage the configuration.

## TABLE OF CONTENTS

<b>Introduction .....</b>	<b>5</b>
<b>Spectrum Scale overview.....</b>	<b>5</b>
Spectrum Scale editions.....	6
E-Series overview.....	6
SANtricity System Manager.....	7
<b>Spectrum Scale installation .....</b>	<b>8</b>
Terminology.....	9
Server operating system requirements .....	9
Running the installation scripts.....	10
Validating the cluster .....	12
<b>InfiniBand validation.....</b>	<b>14</b>
Validating the IB subnet manager.....	14
Install and configure OpenSM .....	14
Using ibping.....	15
Using the ib_ tests .....	15
RDMA status and configuration .....	17
Using nsdperf .....	18
<b>Storage configuration in SANtricity System Manager .....</b>	<b>19</b>
E5760 configuration example with 120 12TB spinning media drives.....	20
EF600 configuration example .....	20
EF600 unallocated capacity required within the volume group.....	21
<b>Tuning Spectrum Scale .....</b>	<b>21</b>
<b>Creating the file system .....</b>	<b>23</b>
Creating udev device rules .....	23
Multipath.conf file.....	24
Creating stanza files .....	25
<b>Validating E-Series data paths through RDMA.....</b>	<b>29</b>
<b>Example testing methodology – CPOC EF600.....</b>	<b>32</b>
Configuration .....	32
EF600 configuration .....	33
NSD configuration file.....	33
File system attributes.....	34

Validating infrastructure configuration with cache reads to the EF600 .....	35
FIO test attributes .....	37
Array performance .....	37
<b>Appendix A: FIO scripts .....</b>	<b>38</b>
<b>Appendix B: mmfslinux.ko error after upgrading OS .....</b>	<b>39</b>
<b>Appendix C: IB port state down .....</b>	<b>39</b>
<b>Appendix D: SuperPod cluster settings .....</b>	<b>41</b>
<b>Appendix E: Setting up tiering .....</b>	<b>41</b>
Prerequisites.....	41
Designate a TCT node class .....	42
Designate cloud service nodes.....	42
Start cloud services .....	43
Define a cloud storage access point.....	43
Create a cloud service .....	43
Create a container pair set .....	43
Create an ILM policy.....	44
<b>Where to find additional information .....</b>	<b>45</b>
<b>Version history.....</b>	<b>45</b>

## LIST OF TABLES

Table 1) Features provided with Spectrum Scale editions.....	6
Table 2) Acronyms. ....	9
Table 3) Options to use with the spectrumscale node add command. ....	11
Table 4) Spectrum Scale parameters. ....	21
Table 5) Spectrum Scale file system configuration parameters.....	34
Table 6) SuperPod cluster attributes. ....	41

## LIST OF FIGURES

Figure 1) Spectrum Scale layout. ....	5
Figure 2) E-Series models.....	7
Figure 3) SANtricity System Manager. ....	8
Figure 4) EF600 volume group layout. ....	21
Figure 5) RDMA statistics.....	30
Figure 6) iSER statistics. ....	31
Figure 7) EF600 RDMA interface. ....	32

Figure 8) CPOC EF600 configuration.....	33
Figure 9) Sequential writes.....	37
Figure 10) Sequential reads. ....	38
Figure 11) Random reads.....	38

## Introduction

Some of the most demanding workloads rely on the ability to expand and contract environments within seconds to minutes. A few workloads, such as Artificial Intelligence (AI), machine learning (ML), and deep learning (DL) capabilities, are rapidly becoming a crucial component for achieving business and scientific breakthroughs. Having robust hardware and software is critical in such environments.

Enter NetApp® E-Series and IBM Spectrum Scale™ solutions. By putting these two proven platforms together, you can have the confidence of nonstop reliability with a robust fault-tolerant design that is easily expandable to meet current and future needs.

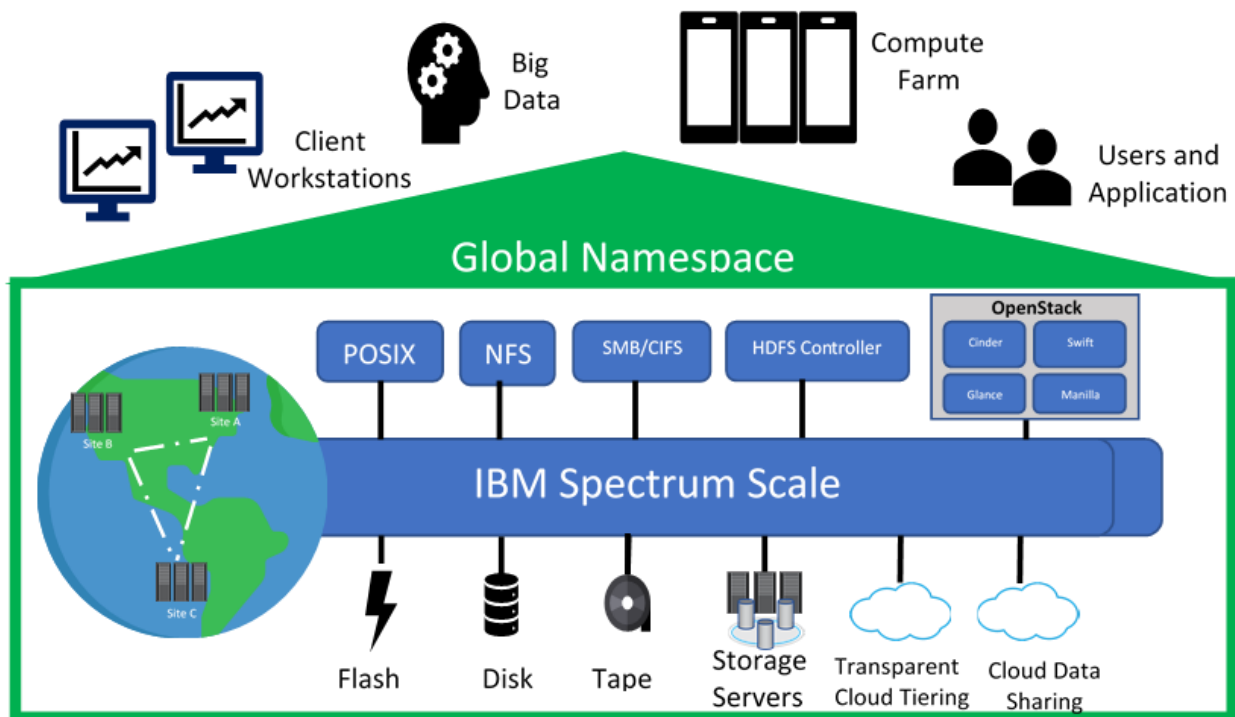
This technical report provides details on how to install, validate, and test a complete Spectrum Scale cluster. It is intended to cover the basic process to deploy f and provide some basic tuning examples. It is not intended to cover extensive performance tuning that might be necessary to match the workloads at customer sites.

## Spectrum Scale overview

Spectrum Scale, formerly General Parallel File System (GPFS), began as a development project in support of the multimedia industry back in 1993. Since its introduction, Spectrum Scale has become one of the premier enterprise class parallel file systems found in many of the top 500 companies.

Businesses and organizations are generating, storing, and analyzing more data than ever before. The race to be the leader in delivering meaningful data processed from rapidly growing data collection is generating the need for enterprise class means to store and retrieve the data. Processing large amounts of big data and artificial intelligence are becoming major initiatives that require robust, secure, and reliable high-performance parallel file systems such as Spectrum Scale, as shown in Figure 1.

Figure 1) Spectrum Scale layout.



With NetApp E-Series storage systems and IBM Spectrum Scale, your HPC workloads will not miss a beat. This proven, integrated solution makes it easy to manage data at scale to help you stay a step ahead of the competition. IBM Spectrum Scale is an enterprise-grade parallel file system that spreads the workload across all storage nodes in a cluster, resulting in accelerated performance and better data access. It offers the distinctive ability to perform archival and analytics in place. Combined with high-performance NetApp E-Series storage, IBM Spectrum Scale delivers industry-leading performance, nonstop availability, and seamless integration with many industry standard protocols.

## Spectrum Scale editions

Editions of Spectrum Scale have varied functional options, as listed in Table 1.

**Table 1) Features provided with Spectrum Scale editions.**

Feature	Data Access	Data Management	Erasure Code
<b>Multi-protocol scalable file service</b> with simultaneous access to a common set of data	√	√	√
<b>Facilitate data access</b> with a global namespace, massively scalable file system, quotas and snapshots, data integrity and availability, and file sets	√	√	√
<b>Simplify management</b> with GUI	√	√	√
<b>Improved efficiency</b> with QoS and compression	√	√	√
<b>Create optimized tiered storage pools</b> based on performance, locality, or cost	√	√	√
<b>Simplify data management</b> with Information Lifecycle Management (ILM) tools that include policy-based data placement and migration	√	√	√
<b>Enable worldwide data access</b> using AFM asynchronous replication	√	√	√
<b>Asynchronous multi-site Disaster Recovery</b>		√	√
<b>Hybrid cloud (TCT)</b>		√	√
<b>Protect data</b> with native software encryption and secure erase, NIST compliant and FIPS certified		√	√
<b>File audit logging</b>		√	√
<b>Watch folder</b>			√
<b>Erasure coding</b>	ESS only	ESS only	√

See the following web site for additional Information regarding editions: [IBM Spectrum Scale Product Editions](#).

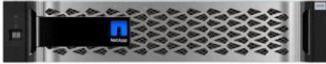




## E-Series overview

The NetApp E-Series high-performance computing (HPC) solution features a complete line of high-performance, highly reliable storage systems. The modular architecture with industry-leading price and performance offers a true pay-as-you-grow solution to support storage requirements for the massive

amounts of data needed for AI, ML, and DL operations. Storage solutions deployed with the leading parallel file system, IBM Spectrum Scale, handle the performance and reliability requirements of the most demanding workloads.

Any model of NetApp E-Series storage system (see Figure 2), along with IBM Spectrum Scale, will give you consistent, near real-time access to your data. IBM Spectrum Scale transparently spreads your data across multiple servers and their back-end storage to prevent bottlenecks and to enable continuous high performance. E-Series storage systems can support any number of workloads simultaneously and can deliver up to two million random read IOPS and 24GBps sustained (maximum burst) write bandwidth per scalable building block. Optimized for both flash and spinning media, E-Series systems include built-in technology that monitors your workloads and automatically adjusts configurations to maximize performance. With the 99.9999% reliability of E-Series systems, your HPC environment will be available whenever and wherever you need it.

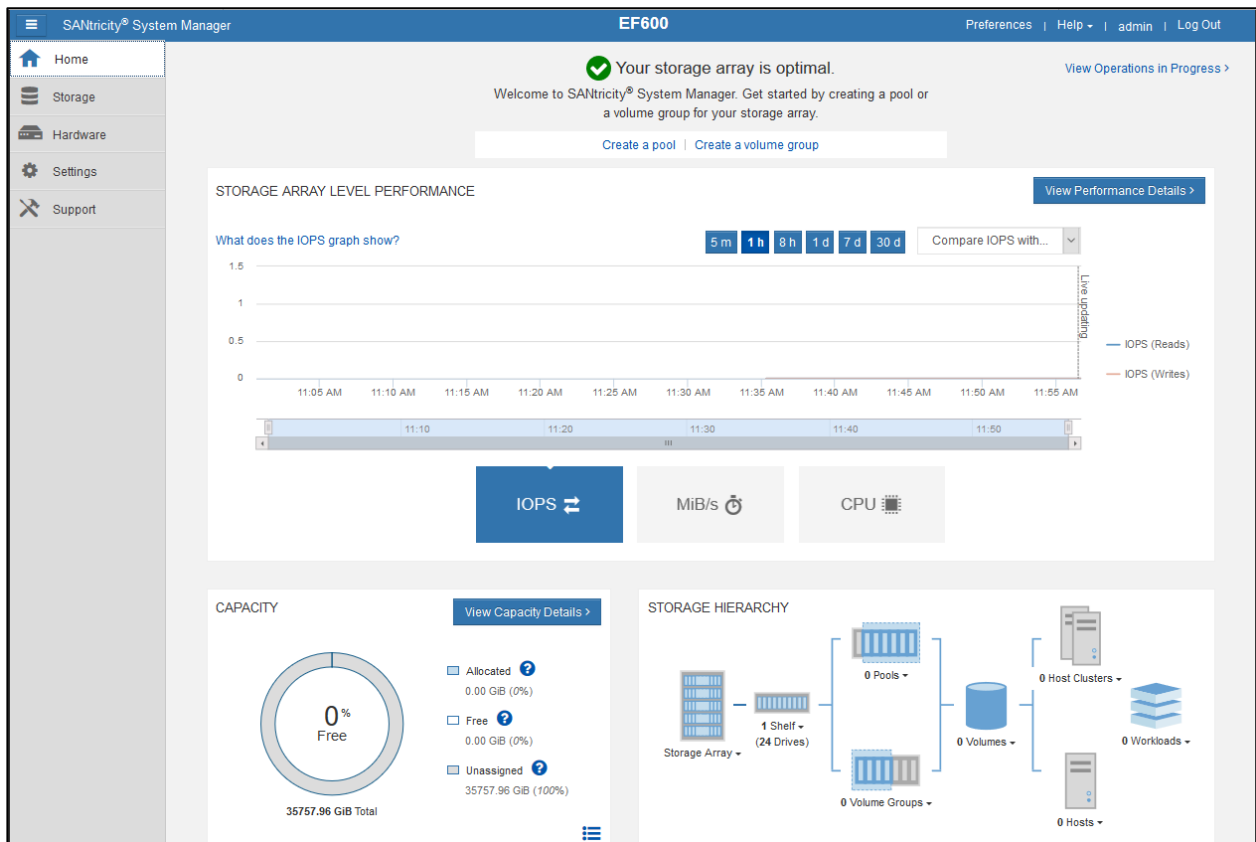
**Figure 2) E-Series models.**

EF600 (End-to-end NVMe)		<ul style="list-style-type: none"> <li>24 NVMe drives</li> <li>NVMe/IB, NVMe/RoCE, NVMe/FC, FC</li> <li>44GBps (sequential reads), 2 million IOPS (random reads)</li> <li>2U24</li> </ul>
EF570 (All-flash array)		<ul style="list-style-type: none"> <li>120 SSD drives</li> <li>FC, SAS, iSCSI, IB, <a href="#">NVMe/IB</a>, <a href="#">NVMe/RoCE</a>, <a href="#">NVMe/FC</a></li> <li>21GBps (sequential reads)</li> <li>1 million IOPS (random reads)</li> </ul>
E5700 (Hybrid flash array)		<ul style="list-style-type: none"> <li>480 drives</li> <li>FC, SAS, iSCSI, IB, <a href="#">NVMe/IB</a>, <a href="#">NVMe/RoCE</a>, <a href="#">NVMe/FC</a></li> <li>21GBps (sequential reads)</li> <li>1 million IOPS (random reads)</li> <li>2U24, 4U60</li> </ul>
EF280 (All-flash array)		<ul style="list-style-type: none"> <li>120 SSD drives</li> <li>FC, SAS, iSCSI</li> <li>21GBps (sequential reads)</li> <li>1 million IOPS (random reads)</li> </ul>
E2800 (Hybrid array)		<ul style="list-style-type: none"> <li>180 drives</li> <li>FC, SAS, iSCSI</li> <li>10GBps (reads)</li> <li>2U12, 2U24, 4U60</li> </ul>

## SANtricity System Manager

With the embedded SANtricity System Manager interface (see Figure 3), you can point any web browser to the array and get up and running quickly. Detailed status, performance, capacity usage, and configuration information are readily available on the initial landing page. Each area in the panel allows for deeper drill down through all layers of the array, including individual drives.

Figure 3) SANtricity System Manager.



## Spectrum Scale installation

IBM provides a Spectrum Scale installation toolkit to automate many of the installation and upgrade tasks. For the toolkit and additional information, see the following topic in the IBM Knowledge Center: [Overview of the Installation Toolkit](#).

The toolkit allows you to automate the following tasks:

- Install and configure Spectrum Scale.
- Perform verification before installing, deploying, or upgrading. It includes checking whether passwordless SSH is set up correctly.
- Add nodes to an existing cluster.
- Add or deploy SMB, NFS, Object, and performance monitoring tools.
- Configure authentication services.
- Upgrade GPFS and all cluster services and install patches.

**Note:** Before version 5.0.4, the installer included both a command line and a graphical user. With 5.0.4 and above, the GUI was deprecated.

The toolkit is in the `~/installation` directory under the bundle extraction location.

```
[root@ictm0803h18-gpfsmaster installer]# pwd
/usr/lpp/mmfs/5.0.4.4/installer
[root@ictm0803h18-gpfsmaster installer]#
```



## Terminology

Table 2 describes acronyms that are used in this document for Spectrum Scale.

**Table 2) Acronyms.**

Acronym	Definition
AFM	Active File Management
CES	Cluster Export Services (NFS, SMB, OBJ)
ESS	Elastic Storage Server (IBM product)
FG	Failure Group
FPO	File Placement Optimizer
GPFS	General Parallel File System (Now Spectrum Scale)
ILM	Information Lifecycle Management
MD	Metadata – Data about data
MM	Multi-Media
NSD	Network Storage Device
NVMe	Nonvolatile Memory express
RDMA	Remote Data Memory Access

## Server operating system requirements

Before running the installer, perform the following steps:

1. Make sure you have the following packages installed:

```
Install Toolkit Dependencies
=====
To make use of the install toolkit the following dependencies must
be met and installed by the end user:

Berkshelf - berks >= 3.1.5
Chef - chef-client, chef-zero & knife >= 12.1.2

Openstack Swift Dependencies
-----
openstack-swift >= 2.7.2

python-eventlet >= 0.18.2
python-greenlet >= 0.4.11

python-keystone >= 9.3.1

python-webob >= 1.2.3
```

2. Set up the `/etc/hosts` file to match each server in the cluster containing all the infrastructure links and IP addresses.

In this example, the primary IP address is on the public ethernet followed by four Infiniband (IB) addresses. This configuration has multiple IB interfaces, but many configurations typically have only have a pair. The NSD servers are directly connected to the storage via IB in this example, but will work just as well via an IB switch.

```
root@dgx2-1:~# cat /etc/hosts
127.0.0.1        localhost
#127.0.1.1       dgx2-1

# The following lines are desirable for IPv6 capable hosts
::1             ip6-localhost ip6-loopback
fe00::0         ip6-localnet
```

```

ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# GPFS Nodes Primary IP
10.61.218.61      nsd-01
10.61.218.62      nsd-02
10.61.218.155     dgx2-1
10.61.218.153     dgx1-3 dgx-3
10.61.218.154     dgx1-4 dgx-4
# Cluster adds IB IP
# NSD 1 and 2
192.168.1.61      nsd-01-p1
192.168.1.71      nsd-01-p3
192.168.1.62      nsd-02-p1
192.168.1.72      nsd-02-p3
# DGX2 Client IB IP
192.168.1.1       dgx2-1-p0
192.168.1.11      dgx2-1-p1
192.168.1.2       dgx2-1-p2
192.168.1.21      dgx2-1-p3
# DGX1-3 Client IB IP
192.168.1.3       dgx1-3-p0
192.168.1.31      dgx1-3-p1
192.168.1.4       dgx1-3-p2
192.168.1.41      dgx1-3-p3
# DGX1-4 Client IB IP
192.168.1.5       dgx1-4-p0
192.168.1.51      dgx1-4-p1
192.168.1.8       dgx1-4-p2
192.168.1.81      dgx1-4-p3

```

### 3. Set up passwordless SSH across all nodes in the cluster, as follows:

```

# ssh-keygen
# ssh-copy-id -i ~/.ssh/id_rsa.pub root@clusterNode

```

4. Depending on your configuration, you can use the IB IP address or the primary IP. Using the IB addresses ensures that all traffic is sent via the IB network. You can set it up using the alias for the IB network listed in the `/etc/hosts` file to give the cluster user-friendly naming. Validate the configuration by running SSH to each node in the cluster. There is a validation check through the installer that will flag any server that is unable to validate, but it might not cover 100% of all routes. Validate it manually to ensure each node can communicate with itself and with each node in the cluster:

- nsd0 > nsd0
- nsd0 > nsd1
- nsd1 > nsd1
- nsd1 > nsd0
- nsd0 > client1
- client1 > client1
- client1 > nsd0
- nsd1 > client1
- client1 > nsd1

## Running the installation scripts

The installer is included with the distribution media.

To run the installer, follow these steps:

1. Open a command line.

2. Define all nodes via the `spectrumscale node add` command. Add options at the end of the command to define the type of node within the cluster.

See Table 3 for a list of arguments to use with the `spectrumscale node add` command.

**Table 3) Options to use with the `spectrumscale node add` command.**

Argument	Definition
-a	Admin node
-m	Manager node
-q	Quorum node
-n	NSD node
-g	Graphic node (must be used with -a)
	No arguments will add node as a client

For more information, see the following topic in the IBM Knowledge Center: [Defining the cluster topology for the installation toolkit](#)

If errors occur during the installation, the installer typically halts with messages about the error, and then logs the location for further diagnosis if necessary. Once you correct the error, you must then re-run the installer so it picks up where it left off.

The following example shows a setup for three clients and two NSD servers. The setup node is the server with the Spectrum Scale bits on it. It technically does not have to be one of the cluster nodes, but it does need access to all of them.

```
nsd-02:/usr/lpp/mmfs/5.0.3.1/installer # ./spectrumscale config gpfs -c cpocail
nsd-02:/usr/lpp/mmfs/5.0.3.1/installer # ./spectrumscale node add dgx1-3-p0 -q
nsd-02:/usr/lpp/mmfs/5.0.3.1/installer # ./spectrumscale node add dgx1-4-p0
nsd-02:/usr/lpp/mmfs/5.0.3.1/installer # ./spectrumscale node add dgx2-1-p0
nsd-02:/usr/lpp/mmfs/5.0.3.1/installer # ./spectrumscale node add nsd-01-p1 -n -q -a
nsd-02:/usr/lpp/mmfs/5.0.3.1/installer # ./spectrumscale node add nsd-02-p1 -n -q -a
nsd-02:/usr/lpp/mmfs/5.0.3.1/installer # ./spectrumscale node list

nsd-02:/usr/lpp/mmfs/5.0.3.1/installer # ./spectrumscale node list
[ INFO ] List of nodes in current configuration:
[ INFO ] [Installer Node]
[ INFO ] 192.168.1.62
[ INFO ]
[ INFO ] [Cluster Details]
[ INFO ] Name: cpocail.dgx1-3-p0
[ INFO ] Setup Type: Spectrum Scale
[ INFO ]
[ INFO ] [Extended Features]
[ INFO ] File Audit logging      : Disabled
[ INFO ] Watch folder           : Disabled
[ INFO ] Management GUI          : Disabled
[ INFO ] Performance Monitoring : Enabled
[ INFO ] Callhome                : Disabled
[ INFO ]
[ INFO ] GPFS      Admin  Quorum  Manager  NSD    Protocol  Perf Mon   OS   Arch
[ INFO ] Node      Node   Node    Node    Server  Node      Collector
[ INFO ] dgx1-3-p0                X                               ubuntu18 x86_64
[ INFO ] dgx1-4-p0                X                               ubuntu18 x86_64
[ INFO ] dgx2-1-p0                X                               ubuntu18 x86_64
[ INFO ] nsd-01-p1      X        X        X        X                               sles12  x86_64
[ INFO ] nsd-02-p1      X        X        X        X                               sles12  x86_64
[ INFO ]
[ INFO ] [Export IP address]
[ INFO ] No export IP addresses configured
nsd-02:/usr/lpp/mmfs/5.0.3.1/installer # ./spectrumscale setup -s 192.168.1.61
nsd-02:/usr/lpp/mmfs/5.0.3.1/installer # ./spectrumscale install --precheck
nsd-02:/usr/lpp/mmfs/5.0.3.1/installer # ./spectrumscale install
```

Any errors from the installation will output along with locations of the log files with additional information.

3. You can run a post installation script to further validate the installation and show additional information on the cluster:

```
nsd-02:./spectrumscale install --postcheck
```

## Validating the cluster

Spectrum Scale provides commands that validate the cluster:

- `mmlscluster`
- `mmhealth cluster show`
- `mmnetverify`

The following examples validate the integrity of all nodes. The default path is `/usr/lpp/mmfs/bin` for the base commands. If you have upgraded or selected a different installation path, the directory might be different. Adding the path to your shell profile makes it much easier for administration.

```
root@dgx2-1:/usr/lpp/mmfs# mmlscluster
```

```
GPFS cluster information
```

```
=====
```

```
GPFS cluster name:      cpocail.dgx1-3-p0
GPFS cluster id:        17383259711231341173
GPFS UID domain:        cpocail.dgx1-3-p0
Remote shell command:   /usr/bin/ssh
Remote file copy command: /usr/bin/scp
Repository type:        CCR
```

Node	Daemon node name	IP address	Admin node name	Designation
1	dgx1-3-p0	192.168.1.3	dgx1-3-p0	quorum-perfmon
2	nsd-01-p1	192.168.1.61	nsd-01-p1	quorum-manager-perfmon
3	nsd-02-p1	192.168.1.62	nsd-02-p1	quorum-manager-perfmon
4	dgx2-1-p0	192.168.1.1	dgx2-1-p0	perfmon
5	dgx1-4-p0	192.168.1.5	dgx1-4-p0	perfmon

To get additional information on any node, add the `-v` option. On an NSD node, it will give information on the disk and volumes:

```
root@dgx2-1:/usr/lpp/mmfs# mmhealth node show -v
```

```
Node name:      dgx2-1-p0
Node status:     TIPS
Status Change:  2020-04-09 14:26:21
```

Component	Status	Status Change	Reasons
GPFS	TIPS	2020-04-09 14:26:21	gpfs_maxstatcache_low
NETWORK	HEALTHY	2020-04-09 14:26:21	-
ib0	HEALTHY	2020-04-09 14:26:21	-
mlx5_6/1	HEALTHY	2020-04-09 14:26:21	-
mlx5_7/1	HEALTHY	2020-04-09 14:26:21	-
mlx5_8/1	HEALTHY	2020-04-09 14:26:21	-
mlx5_9/1	HEALTHY	2020-04-09 14:26:21	-
FILESYSTEM	HEALTHY	2020-04-09 16:45:41	-
cpocgpfs2	HEALTHY	2020-04-09 16:45:41	-
PERFMON	HEALTHY	2020-04-09 14:26:20	-
THRESHOLD	HEALTHY	2020-04-09 14:26:15	-
MemFree_Rule	HEALTHY	2020-04-09 14:27:35	-

```
root@dgx2-1:/usr/lpp/mmfs#
```

From any node, run the `mmnetverify` command to do an extended check on the setup:

```

root@dgx2-1:/usr/lpp/mmfs# mmnetverify

dgx2-1-p0 checking local configuration.
  Operation interface: Success.

dgx2-1-p0 checking communication with node nsd-02-p1.
  Operation resolution: Success.
  Operation ping: Success.
  Operation shell: Success.
  Operation copy: Success.
  Operation time: Success.
  Operation daemon-port: Success.
  Operation sdrserv-port: Success.
  Operation tscmd-port: Success.
  Operation data-small: Success.
  Operation data-medium: Success.

dgx2-1-p0 checking communication with node nsd-01-p1.
  Operation resolution: Success.
  Operation ping: Success.
  Operation shell: Success.
  Operation copy: Success.
  Operation time: Success.
  Operation daemon-port: Success.
  Operation sdrserv-port: Success.
  Operation tscmd-port: Success.
  Operation data-small: Success.
  Operation data-medium: Success.

dgx2-1-p0 checking communication with node dgx1-3-p0.
  Operation resolution: Success.
  Operation ping: Success.
  Operation shell: Success.
  Operation copy: Success.
  Operation time: Success.
  Operation daemon-port: Success.
  Operation sdrserv-port: Success.
  Operation tscmd-port: Success.
  Operation data-small: Success.
  Operation data-medium: Success.

dgx2-1-p0 checking communication with node dgx2-1-p0.
  Operation resolution: Success.
  Operation ping: Success.
  Operation shell: Success.
  Operation copy: Success.
  Operation time: Skipped.
  Operation daemon-port: Success.
  Operation sdrserv-port: Success.
  Operation tscmd-port: Success.
  Operation data-small: Success.
  Operation data-medium: Success.

dgx2-1-p0 checking communication with node dgx1-4-p0.
  Operation resolution: Success.
  Operation ping: Success.
  Operation shell: Success.
  Operation copy: Success.
  Operation time: Success.
  Operation daemon-port: Success.
  Operation sdrserv-port: Success.
  Operation tscmd-port: Success.
  Operation data-small: Success.
  Operation data-medium: Success.

No issues found.
root@dgx2-1:/usr/lpp/mmfs#

```

## InfiniBand validation

Validating the infrastructure is a crucial first step before starting performance testing. InfiniBand (IB) is a predominate interface for parallel file systems.

You can install a testing suite from Mellanox, called Performance Tests, which includes the following tests:

- `ibping`
- `ib_send_bw`
- `ib_send_lat`
- `ib_write_bw`
- `ib_write_lat`
- `ib_read_bw`
- `ib_read_lat`
- `ib_atomic_bw`
- `ib_atomic_lat`

You can find downloads for these tests in the Open Fabrics Distribution (OFED) GitHub page:  
<https://github.com/linux-rdma/perftest>

## Validating the IB subnet manager

InfiniBand (IB) requires the use of a subnet manager that assigns a Local Identifier (LID) to a port connected to an IB network fabric, which puts together a routing table using the LIDs. You can also use hardware and software subnet managers. Hardware subnet managers are typical in a switched environment. The switch has an internal subnet manager that can manage all network connections. If you are not using a switched environment or you have a mixed environment with switch and direct connections, you can install a software package called OpenSM on the server with the IB direct connections to run as the subnet manager. An example is an E-Series array direct attached to the NSD node and additional connections to a switch for the client attachments. In a direct connect environment, when multiple IB connections are set up to the E-Series array, there will be multiple subnet managers running per server, one per connection.

## Install and configure OpenSM

OpenSM is an open source subnet manager that allows you to configure the IB resources when multiple subnet managers are required. You can install the OpenSM package on any hosts that run the subnet manager. Typically, these hosts are the designated NSD servers for a direct connect environment.

The following is an example command for SUSE installations:

```
# zypper install opensm
```

The following is an example command for RHEL/Centos installations:

```
# yum install opensm
Use the ibstat -p command to find GUID0 and GUID1 of the HCA ports. For example:
# ibstat -p
0x248a070300a80a80
0x248a070300a80a81
```

To and install and run the OpenSM package:

1. Start two instances of the subnet manager, one for each subnet, by adding the following commands to `/etc/rc.d/after.local` for SUSE or `etc/rc.d/rc.local` for Red Hat. Substitute the

values you found in the last step for GUID0 and GUID1. For P0 and P1, use the subnet manager priorities, with 1 being the lowest and 15 the highest:

```
opensm -B -g GUID0 -p P0 -f /var/log/opensm-ib0.log
opensm -B -g GUID1 -p P1 -f /var/log/opensm-ib1.log
```

## 2. Validate the entries:

```
# cat /etc/rc.d/rc.local
opensm -B -g 0x248a070300a80a80 -p 15 -f /var/log/opensm-ib0.log
opensm -B -g 0x248a070300a80a81 -p 1 -f /var/log/opensm-ib1.log
```

## 3. Reboot and validate by running the `sminfo` command.

You can find additional information on setup and configuration in the [Linux Express Configuration Guide](#).

## Using ibping

To test initial connectivity, you can use `ibping`, which is a simple server-to-client tool. There are several options to the command line. The following steps show one option for using the tool:

### 1. Start with `ibv_devices` on one of the hosts to list the IB device GUIDs, and then start `ibping` in server mode:

```
[root@ictm0803h11-client1 ~]# ibv_devices
device                node GUID
-----
mlx5_0                98039b030071f768
mlx5_1                98039b030071f769
[root@ictm0803h11-client1 ~]# ibping -S
```

### 2. Open a separate terminal window from one of the hosts in the configuration being tested.

```
[root@ictm0803h09-nsd1 net]# ibping -c 10000 -f -G 0x98039b030071f768

--- ictm0803h11-client1.ict.englab.netapp.com.(none) (Lid 12) ibping statistics ---
10000 packets transmitted, 10000 received, 0% packet loss, time 677 ms
rtt min/avg/max = 0.021/0.067/0.152 ms
[root@ictm0803h09-nsd1 net]#
```

## Using the `ib_` tests

The other `ib` commands are also server-to-client tests, as shown in the following examples. Start the test listener on one node, followed by the test command on the client:

```
[root@ictm0803h09-nsd1 net]# ib_read_bw -R
```

```
*****
* Waiting for client to connect... *
*****
```

```
[root@ictm0803h11-client1 ~]# ib_read_bw -i 1 -R 192.168.100.101
```

```
-----
RDMA_Read BW Test
Dual-port      : OFF      Device      : mlx5_0
Number of qps  : 1        Transport type : IB
Connection type : RC      Using SRQ    : OFF
TX depth       : 128
CQ Moderation  : 100
Mtu            : 4096[B]
Link type      : IB
Outstand reads : 16
rdma_cm QPs    : ON
Data ex. method : rdma_cm
-----
```

```
local address: LID 0x0c QPN 0x012b PSN 0xebe164
```

```

remote address: LID 0x11 QPN 0x0122 PSN 0x81e908
-----
#bytes      #iterations  BW peak[MB/sec]   BW average[MB/sec]   MsgRate[Mpps]
Conflicting CPU frequency values detected: 2653.564000 != 3169.042000. CPU Frequency is not max.
65536       1000         11809.11          11808.57              0.188937
-----
[root@ictm0803h11-client1 ~]#

```

This is the output from the test listener that you started on the first node:

```

[root@ictm0803h09-nsd1 net]# ib_read_bw -R

*****
* Waiting for client to connect... *
*****
-----
RDMA_Read BW Test
Dual-port      : OFF          Device      : mlx5_0
Number of qps  : 1           Transport type : IB
Connection type : RC          Using SRQ      : OFF
CQ Moderation  : 100
Mtu            : 4096[B]
Link type      : IB
Outstand reads : 16
rdma_cm QPs    : ON
Data ex. method : rdma_cm
-----
Waiting for client rdma_cm QP to connect
Please run the same command with the IB/RoCE interface IP
-----
local address: LID 0x11 QPN 0x0122 PSN 0x81e908
remote address: LID 0x0c QPN 0x012b PSN 0xebel64
-----
#bytes      #iterations  BW peak[MB/sec]   BW average[MB/sec]   MsgRate[Mpps]
65536       1000         11809.11          11808.57              0.188937
-----
[root@ictm0803h09-nsd1 net]#

```

Here is a bidirectional test example:

```

[root@ictm0803h09-nsd1 net]# ib_send_bw -R -b

*****
* Waiting for client to connect... *
*****
...

[root@ictm0803h11-client1 ~]# ib_send_bw -i 1 -b -R 192.168.100.101
-----
Send Bidirectional BW Test
Dual-port      : OFF          Device      : mlx5_0
Number of qps  : 1           Transport type : IB
Connection type : RC          Using SRQ      : OFF
TX depth       : 128
RX depth       : 512
CQ Moderation  : 100
Mtu            : 4096[B]
Link type      : IB
Max inline data : 0[B]
rdma_cm QPs    : ON
Data ex. method : rdma_cm
-----
local address: LID 0x0c QPN 0x012d PSN 0x79baad
remote address: LID 0x11 QPN 0x0124 PSN 0x1da379
-----
#bytes      #iterations  BW peak[MB/sec]   BW average[MB/sec]   MsgRate[Mpps]
Conflicting CPU frequency values detected: 3005.566000 != 1200.146000. CPU Frequency is not max.
65536       1000         20562.93          20562.25              0.328996
-----

```



```
[root@ictm0803h11-client1 ~]#
```

This is the output from the test listener that you started on the first node:

```
[root@ictm0803h09-nsd1 net]# ib_send_bw -R -b

*****
* Waiting for client to connect... *
*****

-----
                        Send Bidirectional BW Test
Dual-port      : OFF          Device      : mlx5_0
Number of qps  : 1            Transport type : IB
Connection type : RC          Using SRQ    : OFF
TX depth       : 128
RX depth       : 512
CQ Moderation  : 100
Mtu            : 4096[B]
Link type      : IB
Max inline data : 0[B]
rdma_cm QPs    : ON
Data ex. method : rdma_cm
-----

Waiting for client rdma_cm QP to connect
Please run the same command with the IB/RoCE interface IP
-----

local address: LID 0x11 QPN 0x0124 PSN 0x1da379
remote address: LID 0x0c QPN 0x012d PSN 0x79baad
-----

#bytes    #iterations    BW peak[MB/sec]    BW average[MB/sec]    MsgRate[Mpps]
Conflicting CPU frequency values detected: 2029.687000 != 2915.185000. CPU Frequency is not max.
65536     1000           20562.93           20562.25              0.328996
-----

[root@ictm0803h09-nsd1 net]#
```

## RDMA status and configuration

After Spectrum Scale is running, you must set a few parameters for RDMA mode.

By default, RDMA is disabled in Spectrum Scale, as shown in the following example:

```
[root@ictm0803h18-gpfsmaster ~]# mmlsconfig verbsRdma
verbsRdma disabled
[root@ictm0803h18-gpfsmaster ~]# mmchconfig verbsRdma=enable -N all
[root@ictm0803h18-gpfsmaster ~]# mmlsconfig verbsRdma
verbsRdma enable
```

You must add IB ports within the configuration. If all hosts, NSDs, and clients in the cluster have the same IB configuration, then you can use the `mmchconfig` command as a global command. If there are servers in the cluster that do not have the same configuration, then you use the `-N` option.

```
[root@ictm0803h18-gpfsmaster ~]# mmlsconfig verbsPorts
verbsPorts
[root@ictm0803h18-gpfsmaster ~]# ibstatus
Infiniband device 'mlx5_0' port 1 status:
    default gid:    fe80:0000:0000:0001:9803:9b03:0071:f758
    base lid:       0x11
    sm lid:         0x11
    state:          4: ACTIVE
    phys state:     5: LinkUp
    rate:           100 Gb/sec (4X EDR)
    link_layer:     InfiniBand

Infiniband device 'mlx5_1' port 1 status:
    default gid:    fe80:0000:0000:0001:9803:9b03:0071:f759
    base lid:       0x12
    sm lid:         0x11
    state:          4: ACTIVE
```

```

phys state:      5: LinkUp
rate:           100 Gb/sec (4X EDR)
link_layer:     InfiniBand
[root@ictm0803h18-gpfsmaster ~]# mmchconfig verbsPorts='mlx5_0 mlx5_1
[root@ictm0803h18-gpfsmaster ~]# mmlsconfig verbsPorts
verbsPorts mlx5_0 mlx5_1
[root@ictm0803h18-gpfsmaster ~]#

```

In the following example, the IB ports are not the same on all servers, so using the `-N` option sets them on a per-server configuration. The `nsd-01` and `nsd-02` servers have ports configured as `mlx5_1` and `mlx5_3` for client connections, and the `dgx` servers have `mlx5_1` through 4 configured. The `mmchconfig` command needs to specify the ports that are to be used for RDMA:

```

nsd-02:~ # mmchconfig verbsPorts='mlx5_1 mlx5_3' -N nsd-02-p1
nsd-02:~ # mmchconfig verbsPorts='mlx5_1 mlx5_3' -N nsd-01-p1
nsd-02:~ # mmchconfig verbsPorts='mlx5_0 mlx5_1 mlx5_2 mlx5_3' -N dgx1-4-p0
nsd-02:~ # mmchconfig verbsPorts='mlx5_0 mlx5_1 mlx5_2 mlx5_3' -N dgx1-3-p0
nsd-02:~ # mmchconfig verbsPorts='mlx5_6 mlx5_7 mlx5_8 mlx5_9' -N dgx2-1-p0

```

Another command through Spectrum Scale is `mmfsadm test verbs conn`. This command finds all the RDMA connections between the nodes:

```

[root@ictm0803h18-gpfsmaster net]# mmfsadm test verbs conn
RDMA Connections between nodes:
  destination index cookie fabnum sta num_cli_RD num_cli_WR
-----
gpfsnsd1a      0      1      0 RTS      0      0
gpfsnsd1a      1      2      0 RTS      0      0
gpfsnsd2a      2      3      0 RTS      0      0
gpfsnsd2a      3      4      0 RTS      0      0
gpfsnsd3a      4      5      0 RTS      0      0
gpfsnsd3a      5      6      0 RTS      0      0
gpfsnsd4a      6      7      0 RTS    1588      0
gpfsnsd4a      7      8      0 RTS    1587      0
gpfsclient2a    8      9      0 RTS      0      0
gpfsclient2a    9     10      0 RTS      0      0
gpfsclient3a   10     11      0 RTS      0      0
gpfsclient3a   11     12      0 RTS      0      0
gpfsclient1a   12     13      0 RTS      0      0
gpfsclient1a   13     14      0 RTS      0      0
gpfsclient4a   14     15      0 RTS      0      0
gpfsclient4a   15     16      0 RTS      0      0
[root@ictm0803h18-gpfsmaster net]#

```

## Using nsdperf

IBM has several tools to test a Spectrum Scale cluster, which includes `nsdperf`. Tools like `iperf` are good for testing throughput between two nodes, but trying to use these tools on a larger configuration and coordinating the startup can be difficult. Using `nsdperf` can test the performance of many servers in the cluster simultaneously. It is located in the `/usr/lpp/mmfs/samples/net` directory.

The `nsdperf` tool must be compiled with RDMA support using the `g++` compiler:

```

[root@ictm0803h09-nsd1 net]# cd /usr/lpp/mmfs/samples/net
[root@ictm0803h09-nsd1 net]# ls -lrt
total 608
-r--r--r-- 1 root root 17602 May 28 2019 README
-r--r--r-- 1 root root 266499 May 28 2019 nsdperf.C
-r--r--r-- 1 root root 191 May 28 2019 makefile
[root@ictm0803h09-nsd1 net]# g++ -O2 -DRDMA -o nsdperf-ib -lpthread -lrt -libverbs -lrdmacm
nsdperf.C
[root@ictm0803h09-nsd1 net]# ls -lrt
total 608

```

```
-r--r--r-- 1 root root 17602 May 28 2019 README
-r--r--r-- 1 root root 266499 May 28 2019 nsdperf.C
-r--r--r-- 1 root root 191 May 28 2019 makefile
-rwxr-xr-x 1 root root 325184 Mar 12 15:03 nsdperf-ib
[root@ictm0803h09-nsd1 net]#
```

The easiest way to start the tests is to place the file in a GPFS file system that all servers have access to, so each server runs the same executable.

Each server needs to run `nsdperf` and have the IB cards defined:

```
dgx2-1:/gpfs2/nsdperftest# nsdperf -s -r "mlx5_6:1,mlx5_7:1,mlx5_8:1,mlx5_9:1"
nsd-01:/gpfs2/nsdperftest # nsdperf-rdma -s -r mlx5_1,mlx5_3
nsd-02:/gpfs2/nsdperftest # nsdperf-rdma -s -r mlx5_1,mlx5_3
```

From a separate server, start `nsdperf`. The following example shows some user input:

```
nsdperf> rdma on
RDMA is now on
nsdperf> client dgx2-1-p0
Connected to dgx2-1-p0
nsdperf>
server nsd-01-p1
Connected to nsd-01-p1
nsdperf> server nsd-02-p1
Connected to nsd-02-p1
nsdperf>
nsdperf> test
1-2 nwrite 12100 MB/sec (2890 msg/sec), cli 1% srv 1%, time 10, buff 4194304, RDMA
1-2 read 13100 MB/sec (3120 msg/sec), cli 1% srv 1%, time 10, buff 4194304, RDMA
nsdperf> status
test time: 10 sec
data buffer size: 4194304
TCP socket send/receive buffer size: 0
tester threads: 4
parallel connections: 1
RDMA enabled: yes

clients:
dgx2-1-p0 (192.168.1.1) -> nsd-01-p1 nsd-02-p1
    mlx5_6:1 9803:9b03:0063:8c34
    mlx5_7:1 9803:9b03:0063:8b08
    mlx5_8:1 9803:9b03:0063:83e8
    mlx5_9:1 9803:9b03:0063:8c1c
servers:
nsd-01-p1 (192.168.1.61)
    mlx5_1:1 b859:9f03:0041:8b11
    mlx5_3:1 9803:9b03:0077:436b
nsd-02-p1 (192.168.1.62)
    mlx5_1:1 9803:9b03:0033:e383
    mlx5_3:1 9803:9b03:0033:e3a3
nsdperf> test write
2-1 write 22500 MB/sec (5380 msg/sec), cli 1% srv 1%, time 10, buff 4194304, RDMA
nsdperf> test read
2-1 read 21100 MB/sec (5040 msg/sec), cli 1% srv 2%, time 10, buff 4194304, RDMA
nsdperf>
```

## Storage configuration in SANtricity System Manager

You can choose various methods of configuring storage with Spectrum Scale, including the RAID type to use, the number of drives in a volume group, and whether to use separate metadata volumes or combined metadata and data volumes. Here are a few suggestions:

- Create a traditional configuration, which includes RAID 6 (8+2) volume groups for data, separate RAID 1 volume groups for the metadata, and one volume per volume group. Optimally, you should

have the metadata on SSD drives, which offers the best performance and scalability option as long as there are enough volume groups/volumes to balance the workload to the hosts.

- If all drives are the same, create volume groups as RAID 6 (8+2), and volumes to combine metadata and data on all volumes. Depending on workloads, this configuration might not provide optimal performance.
- Configure disk pools to create multiple volumes to balance the workload on the server. However, depending on the workload, disk pools might not offer optimal performance. Disk pools can be beneficial to reduce rebuild times when there are drive failures.
- Disable dynamic cache read prefetch. Spectrum Scale has its own algorithm for this feature, and disabling it within the E-Series array might improve performance.
- Choose the segment size applicable for your workload. Large file writes might benefit from 512KiB while smaller writes might benefit from 128KiB or 256KiB segment size.
- Create a cluster group for assigning hosts and volumes, so every host has access to the volumes.

## **E5760 configuration example with 120 12TB spinning media drives**

On the E5760 storage array, testing has shown that configuring 10-drive (8+2) RAID 6 volume groups optimizes for the best overall and consistent performance. Adding a set of SSDs for metadata in a RAID 1 configuration can optimize performance.

With 120 drives in the array, there would be 12 volume groups created in 8+2 configurations and depending on how many NSD servers are attached to the array, creating one volume per NSD server. Each volume group would have one volume created and mapped to each NSD server to spread out the load evenly. The number of RAID 1 groups needed will depend on the workload. Two RAID 1 volume groups with a single volume mapped to the host is a good starting place.

## **EF600 configuration example**

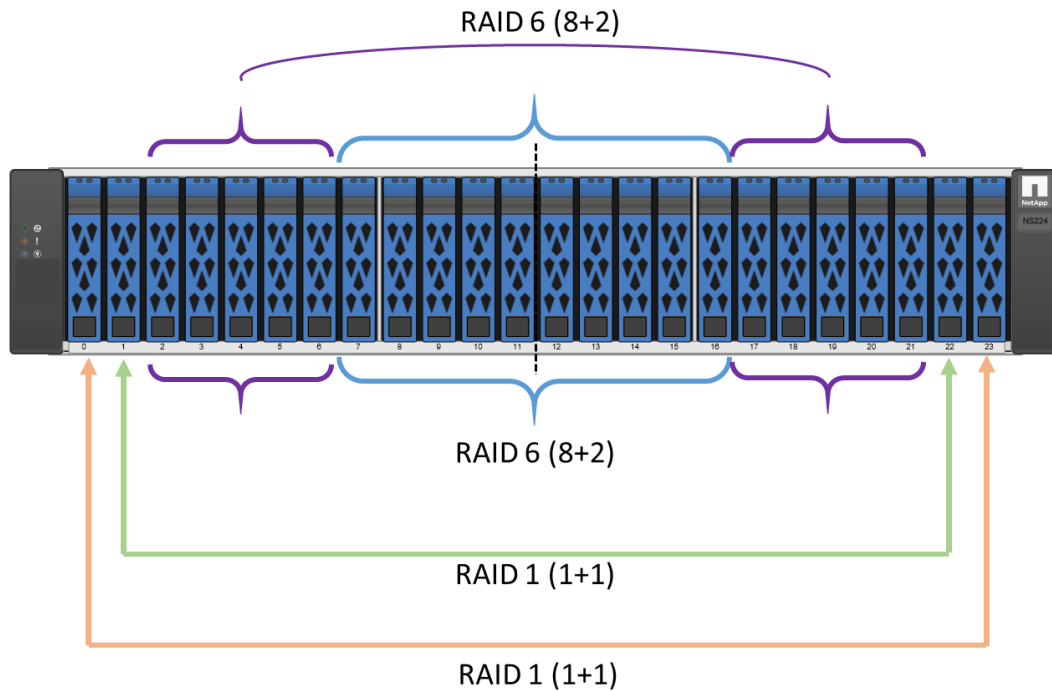
With 24 drives, the EF600 array offers several configuration options:

- Create two RAID 6 (8+2) volume groups for data and two RAID 1 volume groups for metadata.
- Create a 10+2 RAID 6 configuration with two volume groups, and four volumes per group. This configuration will allow for balancing across both HICs per controller.

For balancing the workload across EF600 internal PCI busses, consider the following:

- With two HICs on each controller, make sure that connections are balanced across both HICs and that you use enough ports to reach controller bandwidth limits.
  - For FC, all ports must be used to reach controller bandwidth limits.
  - For EDR IB/ROCE, at least one port per HIC is needed.
- Drive slot assignment is different with the EF600 array, as opposed to the EF570 array. The EF570 alternates slots, whereas the EF600 has the first PCIe bus connected to the drive slots 0 to 11, first 12 drive slots, and the second PCIe bus is connected to drives in slots 12 through 23, second 12 slots.
  - Each controller should have access to an equal number of drives in the first 12 slots, and from the last 12 slots to use both drive side PCIe busses effectively. Figure 4 shows examples where RAID 6 volume groups are created from the middle drives out and RAID 1 drives are built from the outside in. Currently SANtricity allows for drive selection under the Advanced feature when creating a volume group. For disk pool creation, if you are not using all drives, the REST API can be used to select the drives for the disk pool group.

Figure 4) EF600 volume group layout.



## EF600 unallocated capacity required within the volume group

When performing write intensive testing on an EF600 or when configuring for write-intensive workloads, review the follow article: [Knowledge Base article H05527](#) – “How to increase EF600 SSD endurance and maximize write performance by reserving free capacity.”

The article recommends leaving unallocated space in a volume group or disk pool group that will result in write performance consistency.

## Tuning Spectrum Scale

The release of Spectrum Scale 5.0.x introduced significant enhancements in performance, performance tuning, and administration automation. These updates have made installation and setup of a full cluster an easy and streamlined process. By using the installation toolkit provided by Spectrum Scale, you can validate all pre-requisites and system configuration requirements before installation. Once installed, the default configuration parameters typically produce acceptable results. You should consider each tuning parameter, as shown in Table 4. All parameters need to be reviewed and validated for each configuration.

To view current settings, use the `mmlsconfig` command. To change the setting, use the `mmchconfig` command.

Table 4) Spectrum Scale parameters.

Parameter	Default	Suggested
ignorePrefetchLUNCount	No	Yes (per NSD)
maxblocksize	4M	16M
maxBufferDescs	32k	2m

Parameter	Default	Suggested
maxFilesToCache	6000	1m
maxInodeDeallocHistory	50	0
maxMBpS	100000	40000 (configuration dependent)
maxStatCache	10000	2m
nsdbufspace	70	70
nsdMaxWorkerThreads	1024	3k
nsdMinWorkerThreads	8	3k
pagepool	2G	96g (per NSD)
scatterBufferSize	262144	256k
verbsPorts	-	Configuration dependent
verbsRdma	Disable	Enable
verbsRdmaMinBytes	4096	32k
verbsRdmaSend	no	Yes
workerThreads	512	1024

Each parameter can be individually displayed or can be on a single line with comma delimiters.

```
[root@ictm0803h09-nsd1 mmfs]# mmfscfg ignorePrefetchLUNCount
ignorePrefetchLUNCount no
[root@ictm0803h09-nsd1 mmfs]# mmfscfg maxblocksize
maxblocksize 16M
[root@ictm0803h09-nsd1 mmfs]# mmfscfg maxBufferDescs
maxBufferDescs 32k
[root@ictm0803h09-nsd1 mmfs]# mmfscfg maxFilesToCache
maxFilesToCache 60000
[root@ictm0803h09-nsd1 mmfs]# mmfscfg maxInodeDeallocHistory
maxInodeDeallocHistory 50
[root@ictm0803h09-nsd1 mmfs]# mmfscfg maxMBpS
maxMBpS 100000
[root@ictm0803h09-nsd1 mmfs]# mmfscfg maxStatCache
maxStatCache 10000
[root@ictm0803h09-nsd1 mmfs]# mmfscfg nsdbufspace
nsdbufspace 70
[root@ictm0803h09-nsd1 mmfs]# mmfscfg nsdMaxWorkerThreads
nsdMaxWorkerThreads 1024
[root@ictm0803h09-nsd1 mmfs]# mmfscfg nsdMinWorkerThreads
nsdMinWorkerThreads 8
[root@ictm0803h09-nsd1 mmfs]# mmfscfg pagepool
pagepool 2G
[root@ictm0803h09-nsd1 mmfs]# mmfscfg scatterBufferSize
scatterBufferSize 262144
[root@ictm0803h09-nsd1 mmfs]# mmfscfg verbsRdma
verbsRdma enable
[root@ictm0803h09-nsd1 mmfs]# mmfscfg verbsRdmaMinBytes
verbsRdmaMinBytes 4096
[root@ictm0803h09-nsd1 mmfs]# mmfscfg verbsRdmaSend
verbsRdmaSend yes
[root@ictm0803h09-nsd1 mmfs]# mmfscfg workerThreads
workerThreads 512
```

Changing the parameters can be done individually or on a space-separated command line. After modifications, you should restart Spectrum Scale.

```
mmchconfig ignorePrefetchLUNCount=yes -N gpfsnsd1a,gpfsnsd2a,gpfsnsd3a,gpfsnsd4a maxblocksize=16M
maxBufferDescs=2m maxFilesToCache=1m,maxInodeDeallocHistory=0 maxMBpS=40000 maxStatCache=2m
nsdbufspace=70 nsdMaxWorkerThreads=3k nsdMinWorkerThreads=3k pagepool=96g -N
gpfsnsd1a,gpfsnsd2a,gpfsnsd3a,gpfsnsd4a scatterBufferSize=256k verbsRdma=enable
verbsRdmaMinBytes=32k verbsRdmaSend=yes workerThreads=1024
mmchconfig: Command successfully completed
```

```
mmchconfig: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
[root@ictm0803h09-nsd1 GPFS-bits]# mmshutdown -a
Tue Jun  9 09:04:57 CDT 2020: mmshutdown: Starting force unmount of GPFS file systems
Tue Jun  9 09:05:07 CDT 2020: mmshutdown: Shutting down GPFS daemons
Tue Jun  9 09:05:24 CDT 2020: mmshutdown: Finished
[root@ictm0803h09-nsd1 GPFS-bits]# mmstartup -a
Tue Jun  9 09:05:32 CDT 2020: mmstartup: Starting GPFS ...
```

Next, you must validate the changes, as follows:

```
[root@ictm0803h09-nsd1 GPFS-bits]# mmlsconfig
ignorePrefetchLUNCount,maxblocksize,maxBufferDescs,maxFilesToCache,maxInodeDeallocHistory,maxMBpS
,maxStatCache,nsdbufspace,nsdMaxWorkerThreads,nsdMinWorkerThreads,pagepool,scatterBufferSize,verb
sRdma,verbsPorts,verbsRdmaMinBytes,verbsRdmaSend,workerThreads
ignorePrefetchLUNCount no
ignorePrefetchLUNCount yes [gpfsnsd1a,gpfsnsd2a,gpfsnsd3a,gpfsnsd4a]
maxblocksize 16M
maxBufferDescs 32k
maxFilesToCache 60000
maxInodeDeallocHistory 50
maxMBpS 100000
maxStatCache 10000
nsdbufspace 70
nsdMaxWorkerThreads 1024
nsdMinWorkerThreads 8
pagepool 2G
pagepool 96g [gpfsnsd1a,gpfsnsd2a,gpfsnsd3a,gpfsnsd4a]
scatterBufferSize 262144
verbsRdma enable
verbsPorts mlx5_0 mlx5_1
verbsRdmaMinBytes 4096
verbsRdmaSend yes
workerThreads 1024
[root@ictm0803h09-nsd1 GPFS-bits]#
```

## Creating the file system

To optimize Spectrum Scale, you must configure several parameters on the NSD servers by performing these tasks:

- Create a `udev` device rules table
- Update the `/etc/multipath.conf` file
- Create the `stanza` file

You should also map array volumes to the cluster group in SANtricity System Manager, and then show them at the NSD server under the Linux multipath driver. By default, the devices are discovered from the multipath drivers at the NSD hosts; however, you must add several parameters to optimize Spectrum Scale.

## Creating udev device rules

Linux schedulers can optimize disk access by spinning media to get better performance. The three most common schedulers are Noop, Deadline, and CFQ, each having their pros and cons. For E-Series systems with Spectrum Scale, we recommend setting the scheduler to Noop. This scheduler uses a first-in, first-out approach queuing data. In addition, a `udev` rules file will persist scheduler changes across boots. You should set the `read_ahead_kb` to match the segment size in SANtricity when you created the volumes.

The following example shows how to configure a `udev` rule, although there are many ways of creating `udev` rules. This example should work for testing purposes. You must adjust variables to your setup when using mixed media, such as SSDs and HDDs.

```
[root@ictm0803h09-nsd1 rules.d]# cat /etc/udev/rules.d/86-gpfs-netapp-io-scheduler.rules
# set readahead size
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/read_ahead_kb}="256"
ACTION=="add|change", KERNEL=="sda[a-z]", ATTR{queue/read_ahead_kb}="256"
ACTION=="add|change", KERNEL=="dm-*", ATTR{queue/read_ahead_kb}="256"

# set noop scheduler
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/scheduler}="noop"
ACTION=="add|change", KERNEL=="sda[a-z]", ATTR{queue/scheduler}="noop"
ACTION=="add|change", KERNEL=="dm-*", ATTR{queue/scheduler}="noop"
```

Be sure to validate the rules. The running scheduler will show in the brackets [].

```
[root@ictm0803h09-nsd1 rules.d]# udevadm control --reload-rules
[root@ictm0803h09-nsd1 rules.d]# udevadm trigger --type=devices --action=change
[root@ictm0803h09-nsd1 rules.d]# cat /sys/block/sdad/queue/scheduler
[noop] deadline cfq
[root@ictm0803h09-nsd1 rules.d]# cat /sys/block/dm-4/queue/scheduler
[noop] deadline cfq
[root@ictm0803h09-nsd1 rules.d]# cat /sys/block/dm-4/queue/read_ahead_kb
256
[root@ictm0803h09-nsd1 rules.d]#
```

## Multipath.conf file

In general, the devices are discovered and configured automatically with the default multipath. You can set a few parameters to help with performance, such as `max_sectors_kb`. All the other entries can be omitted if you are using any of the latest Linux versions, as they are set in the kernel for E-Series devices.

You should set the `max_sectors_kb` to the same segment size used when you created the volume in SANtricity System Manager. This setting keeps the sector alignment and improves performance.

You can also configure aliases in the `/etc/multipath.conf` file for each of the E-Series volumes to make the mapping of the `/dev/dm-*` device easier from each of the NSD servers.

The following example shows a `multipath.conf` file configuration.

```
[root@ictm0803h09-nsd1 ~]# cat /etc/multipath.conf
defaults {
    user_friendly_names    yes
    find_multipaths        yes
}
devices {
    device {
        vendor              "NETAPP"
        product             "INF-01-00"
        product_blacklist   "Universal Xport"
        path_grouping_policy "group_by_prio"
        path_checker        "rdac"
        features            "2 pg_init_retries 50"
        hardware_handler    "1 rdac"
        prio                "rdac"
        failback            "immediate"
        rr_weight           "uniform"
        no_path_retry       30
        retain_attached_hw_handler yes
        detect_prio        yes
        max_sectors_kb     2048
    }
}
multipaths {
    multipath {
        wwid                3600a098000bffd208000019045e3962db
        alias               GPFS-SSD-Meta-V1a
    }
    multipath {
        wwid                3600a098000bffd1ae00001b805e396317
        alias               GPFS-SSD-Meta-V1b
    }
}
```



```

    }
    multipath {
        wwid                      3600a098000bffd208000019075e3962e1
        alias                     GPFS-SSD-Meta-V1c
    }
    multipath {
        wwid                      3600a098000bffd1ae00001b825e39631d
        alias                     GPFS-SSD-Meta-V1d
    }
    ...
[root@ictm0803h09-nsd1 ~]# multipath -ll |grep NET |sort
GPFS-SSD-V1a (3600a098000bffd208000018fa5e3962c4) dm-10 NETAPP ,INF-01-00
GPFS-SSD-V1b (3600a098000bffd1ae00001b785e396300) dm-12 NETAPP ,INF-01-00
GPFS-SSD-V1c (3600a098000bffd208000018fd5e3962ca) dm-14 NETAPP ,INF-01-00
GPFS-SSD-V1d (3600a098000bffd1ae00001b7a5e396306) dm-17 NETAPP ,INF-01-00

```

## Creating stanza files

You can create a stanza file that contains the device name for each volume. Spectrum Scale uses entries in the `/dev` directory to map volumes to NSD devices. For multipathing, you can use the `/dev/dm-*` devices. A mapping issue might occur if the devices are not the same across the NSD servers. You can configure the stanza file to assign alternating volumes to the NSD servers to spread out balance performance as well as create High Availability failover. Each device entry needs to map to the first entry in the server's field. The second entry is the backup or alternate server entry.

In the stanza file, you can create separate volumes for `metadataOnly` and separate volumes to store data only. For example, you could create a set of SSD volumes for metadata, along with a set of spinning HDDs for the data volumes. Alternately if all the drives are the same capacity and type, and all spinning media for example, they can all be configured as `dataAndMetadata` volumes to spread the data and performance across all the drives.

```

%nsd: device=DiskName
    nsd=NsdName
    servers=ServerList
    usage={dataOnly | metadataOnly | dataAndMetadata | descOnly | localCache}
    failureGroup=FailureGroup

```

The following example uses data and metadata on the same volume. The device name correlates to the volume on the first NSD server and is listed first on the server's line. This example is using two NSD servers where the volumes will alternate primary server for volume access and the secondary or alternate server second in the list.

```

%nsd: device=/dev/dm-3
nsd=VOLUME_001_ssd_v1a
servers=gpfsnsd1a,gpfsnsd2a
usage=dataAndMetadata
pool=system
failureGroup=1

%nsd: device=/dev/dm-4
nsd=VOLUME_002_ssd_v1b
servers=gpfsnsd2a,gpfsnsd1a
usage=dataAndMetadata
pool=system
failureGroup=1

```

The following example shows separate metadata and data volumes. One stanza file is created to assign the volumes to Spectrum Scale.

```

[root@ictm0803h09-nsd1 mmfs]# cat StanzaFile-SSD-HDD-v1
%nsd: device=/dev/dm-3
nsd=VOLUME_001_SSD_META_v1
servers=gpfsnsd1a

```

```

usage=metadataOnly
pool=system
failureGroup=1

%nsd: device=/dev/dm-4
nsd=VOLUME_002_SSD_META_v2
servers=gpfsnsd1a
usage=metadataOnly
pool=system
failureGroup=2

%nsd: device=/dev/dm-5
nsd=VOLUME_003_HDD_Data_V1
servers=gpfsnsd1a
usage=dataOnly
pool=TME_Pool2
failureGroup=3

%nsd: device=/dev/dm-6
nsd=VOLUME_004_HDD_Data_V2
servers=gpfsnsd1a
usage=dataOnly
pool=TME_Pool2
failureGroup=3

%nsd: device=/dev/dm-7
nsd=VOLUME_003_HDD_Data_V3
servers=gpfsnsd1a
usage=dataOnly
pool=TME_Pool2
failureGroup=3

%nsd: device=/dev/dm-8
nsd=VOLUME_004_HDD_Data_V4
servers=gpfsnsd1a
usage=dataOnly
pool=TME_Pool2
failureGroup=3

%nsd: device=/dev/dm-10
nsd=VOLUME_003_HDD_Data_V5
servers=gpfsnsd1a
usage=dataOnly
pool=TME_Pool2
failureGroup=4

%nsd: device=/dev/dm-9
nsd=VOLUME_004_HDD_Data_V6
servers=gpfsnsd1a
usage=dataOnly
pool=TME_Pool2
failureGroup=4

%nsd: device=/dev/dm-11
nsd=VOLUME_003_HDD_Data_V7
servers=gpfsnsd1a
usage=dataOnly
pool=TME_Pool2
failureGroup=4

%nsd: device=/dev/dm-12
nsd=VOLUME_004_HDD_Data_V8
servers=gpfsnsd1a
usage=dataOnly
pool=TME_Pool2
failureGroup=4

```

A second file is created and used to re-assign primary NSD servers and assign backups. With the second file, you do not have to map the `dm` devices from other servers, as they might not all be the same.

```

[root@ictm0803h09-nsd1 mmfs]# cat StanzaFile-SSD-HDD-v1-balance
%nsd: device=/dev/dm-3
nsd=VOLUME_001_SSD_META_v1
servers=gpfsnsd1a,gpfsnsd2a
usage=metadataOnly
pool=system
failureGroup=1

%nsd: device=/dev/dm-4
nsd=VOLUME_002_SSD_META_v2
servers=gpfsnsd2a,gpfsnsd1a
usage=metadataOnly
pool=system
failureGroup=2

%nsd: device=/dev/dm-5
nsd=VOLUME_003_HDD_Data_V1
servers=gpfsnsd2a,gpfsnsd1a
usage=dataOnly
pool=TME_Pool2
failureGroup=3

%nsd: device=/dev/dm-6
nsd=VOLUME_004_HDD_Data_V2
servers=gpfsnsd1a,gpfsnsd2a
usage=dataOnly
pool=TME_Pool2
failureGroup=3

%nsd: device=/dev/dm-7
nsd=VOLUME_005_HDD_Data_V3
servers=gpfsnsd1a,gpfsnsd2a
usage=dataOnly
pool=TME_Pool2
failureGroup=3

%nsd: device=/dev/dm-8
nsd=VOLUME_006_HDD_Data_V4
servers=gpfsnsd2a,gpfsnsd1a
usage=dataOnly
pool=TME_Pool2
failureGroup=3

%nsd: device=/dev/dm-10
nsd=VOLUME_007_HDD_Data_V5
servers=gpfsnsd2a,gpfsnsd1a
usage=dataOnly
pool=TME_Pool2
failureGroup=4

%nsd: device=/dev/dm-9
nsd=VOLUME_008_HDD_Data_V6
servers=gpfsnsd1a,gpfsnsd2a
usage=dataOnly
pool=TME_Pool2
failureGroup=4

%nsd: device=/dev/dm-11
nsd=VOLUME_009_HDD_Data_V7
servers=gpfsnsd1a,gpfsnsd2a
usage=dataOnly
pool=TME_Pool2
failureGroup=4

%nsd: device=/dev/dm-12
nsd=VOLUME_010_HDD_Data_V8
servers=gpfsnsd1a,gpfsnsd2a
usage=dataOnly
pool=TME_Pool2
failureGroup=4

```

```
[root@ictm0803h09-nsd1 mmfs]#
```

The following example commands are for creating the NSDs using the initial stanza file, then running the second file against the file system to assign primary and secondary hosts.

```
[root@ictm0803h09-nsd1 mmfs]# mmcrnsd -F ././StanzaFile-SSD-HDD-v1
mmcrnsd: Processing disk dm-3
mmcrnsd: Processing disk dm-4
mmcrnsd: Processing disk dm-5
mmcrnsd: Processing disk dm-6
mmcrnsd: Processing disk dm-7
mmcrnsd: Processing disk dm-8
mmcrnsd: Processing disk dm-10
mmcrnsd: Processing disk dm-9
mmcrnsd: Processing disk dm-11
mmcrnsd: Processing disk dm-12
mmcrnsd: Propagating the cluster configuration data to all
        affected nodes. This is an asynchronous process.
[root@ictm0803h09-nsd1 mmfs]# mmchnsd -F StanzaFile-SSD-HDD-v1-balance
mmchnsd: Processing disk VOLUME_001_SSD_META_v1
mmchnsd: Processing disk VOLUME_002_SSD_META_v2
mmchnsd: Processing disk VOLUME_003_HDD_Data_V1
mmchnsd: Processing disk VOLUME_004_HDD_Data_V2
mmchnsd: Processing disk VOLUME_005_HDD_Data_V3
mmchnsd: Processing disk VOLUME_006_HDD_Data_V4
mmchnsd: Processing disk VOLUME_007_HDD_Data_V5
mmchnsd: Processing disk VOLUME_008_HDD_Data_V6
mmchnsd: Processing disk VOLUME_009_HDD_Data_V7
mmchnsd: Processing disk VOLUME_010_HDD_Data_V8
mmchnsd: Propagating the cluster configuration data to all
        affected nodes. This is an asynchronous process.
[root@ictm0803h09-nsd1 mmfs]# mmlnsd
```

File system	Disk name	NSD servers
(free disk)	VOLUME_001_SSD_META_v1	gpfsnsd1a.ict.englab.netapp.com, gpfsnsd2a.ict.englab.netapp.com
(free disk)	VOLUME_002_SSD_META_v2	gpfsnsd2a.ict.englab.netapp.com, gpfsnsd1a.ict.englab.netapp.com
(free disk)	VOLUME_003_HDD_Data_V1	gpfsnsd2a.ict.englab.netapp.com, gpfsnsd1a.ict.englab.netapp.com
(free disk)	VOLUME_004_HDD_Data_V2	gpfsnsd1a.ict.englab.netapp.com, gpfsnsd2a.ict.englab.netapp.com
(free disk)	VOLUME_005_HDD_Data_V3	gpfsnsd1a.ict.englab.netapp.com, gpfsnsd2a.ict.englab.netapp.com
(free disk)	VOLUME_006_HDD_Data_V4	gpfsnsd2a.ict.englab.netapp.com, gpfsnsd1a.ict.englab.netapp.com
(free disk)	VOLUME_007_HDD_Data_V5	gpfsnsd2a.ict.englab.netapp.com, gpfsnsd1a.ict.englab.netapp.com
(free disk)	VOLUME_008_HDD_Data_V6	gpfsnsd1a.ict.englab.netapp.com, gpfsnsd2a.ict.englab.netapp.com
(free disk)	VOLUME_009_HDD_Data_V7	gpfsnsd1a.ict.englab.netapp.com, gpfsnsd2a.ict.englab.netapp.com
(free disk)	VOLUME_010_HDD_Data_V8	gpfsnsd1a.ict.englab.netapp.com, gpfsnsd2a.ict.englab.netapp.com

```
[root@ictm0803h09-nsd1 mmfs]#
```

Spectrum Scale will mount the file system and create entries into `/etc/fstab` when the file system gets created through Spectrum Scale.

The next task is to build the file system using the same stanza file. You can apply options to `mmcrfs`, such as `-B` to specify the block size. If you are testing a large number of smaller files, you should consider the `-inode-limit`. Mounting the file system with `mmmount` makes the file system available for use on all nodes.

```
# mmcrfs gpfs1 -F ././StanzaFile-SSD-HDD-v1-balance -B 16M -inode-limit 100000000:10000000 -T
/gpfs1
[root@ictm0803h09-nsd1 mmfs]# mmmount gpfs1 -a
```

```

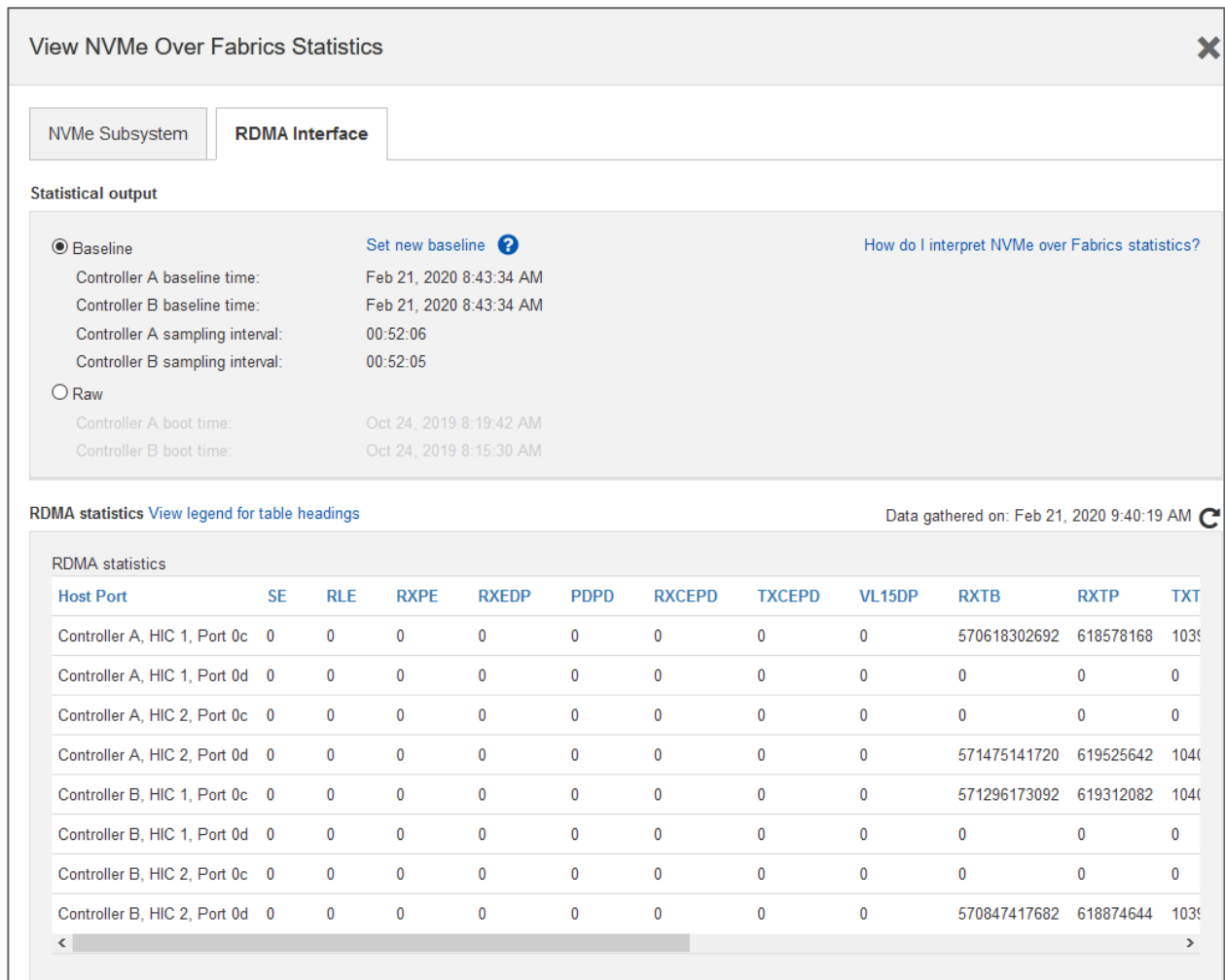
Mon Jun  8 15:58:05 CDT 2020: mmmount: Mounting file systems ...
[root@ictm0803h09-nsd1 mmfs]# df
Filesystem                                1K-blocks      Used    Available Use% Mounted on
devtmpfs                                  32597308         0     32597308   0% /dev
tmpfs                                     32611884         0     32611884   0% /dev/shm
tmpfs                                     32611884    19272     32592612   1% /run
tmpfs                                     32611884         0     32611884   0% /sys/fs/cgroup
/dev/mapper/rootvg-root                   112437616 24595196     87842420  22% /
/dev/sda1                                 505580      168988       336592  34% /boot
tmpfs                                     6522380         0     6522380   0% /run/user/0
10.113.251.170:/pxe/applications          722347264 430782976     254871296  63% /mnt/apps
gpfs1                                     611873456128 786432 611872669696   1% /gpfs1
[root@ictm0803h09-nsd1 mmfs]#

```

## Validating E-Series data paths through RDMA

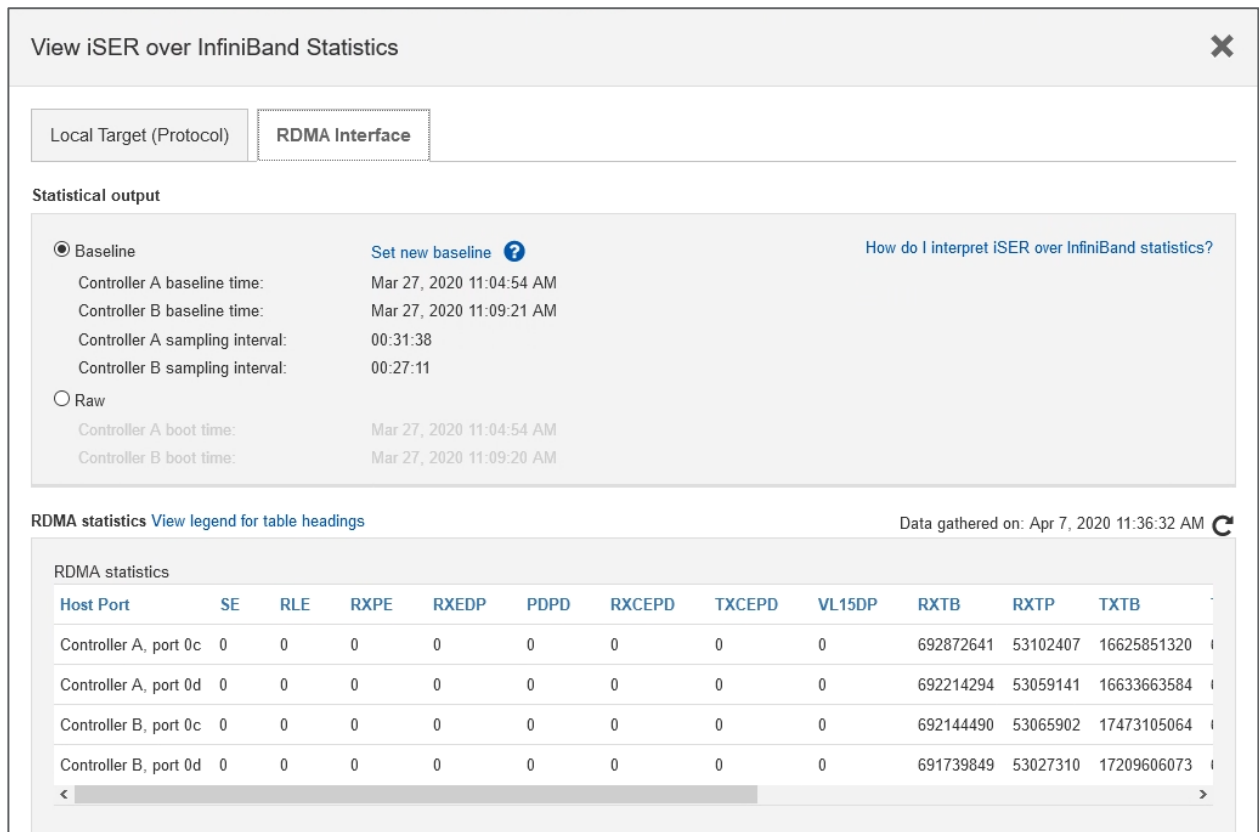
SANtricity System Manager offers an easy way to validate that the Spectrum Scale file system and NSD assignments are working correctly and utilizing all configured data paths. In the System Manager interface, go to the Settings > System > View NVMe over Fabrics Statistics page, and then select the RDMA Interface tab, which shows all active paths (see Figure 5). If it does not show the typical four paths running, it might be due to the way the stanza file and the preferred path for volumes are set up. A possible workaround is to assign two volumes to the A controller, then the next two volumes to the B controller as the preferred path, as opposed to alternating every other one.

Figure 5) RDMA statistics.



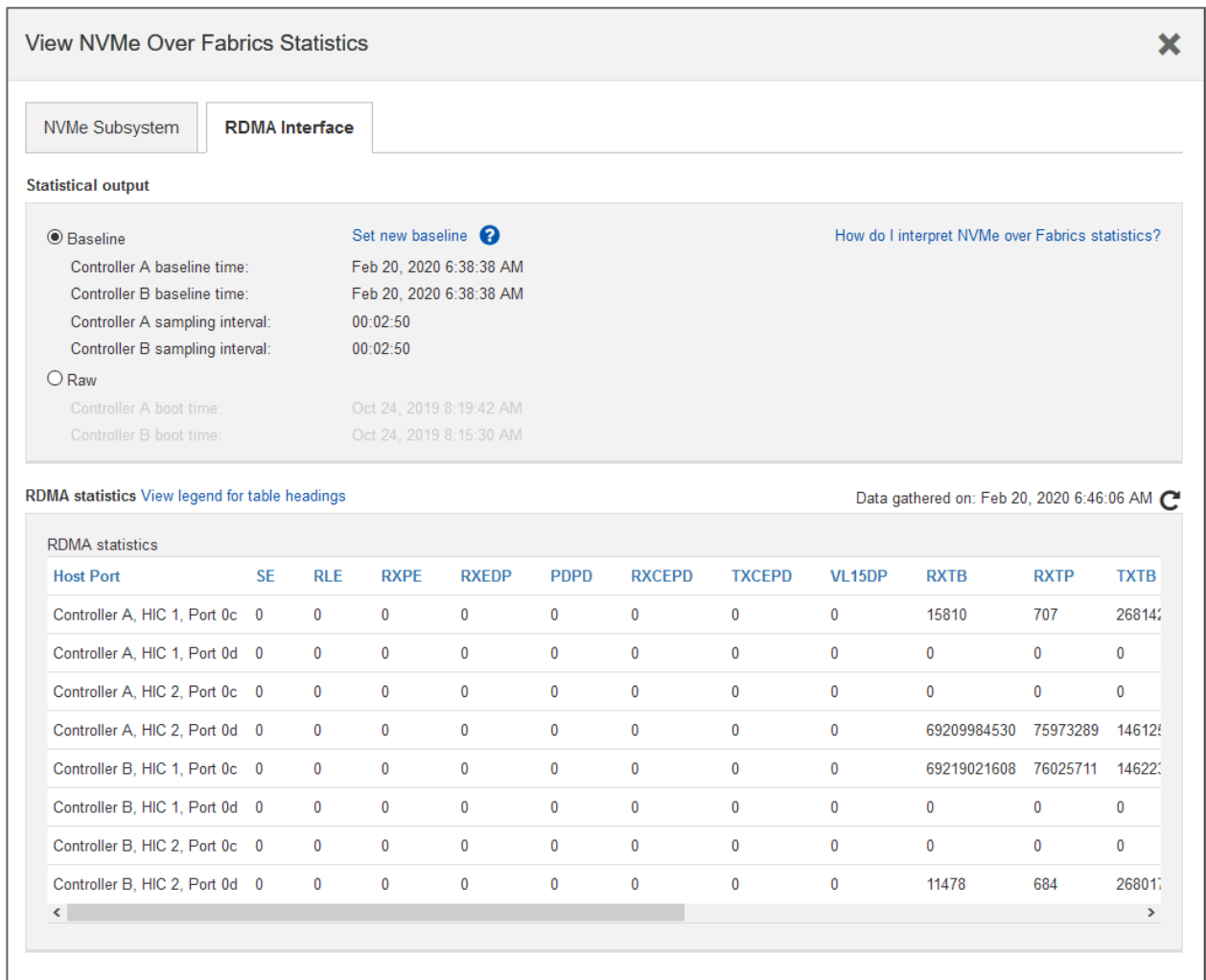
You can find the same page (see Figure 6), if you are using iSER over InfiniBand on an E5700.

Figure 6) iSER statistics.



There are cases where two rights can make a wrong. Figure 7 shows an EF600 where two NSD servers are connected, and very little IO can be seen on two of the ports. This was a case where mapping every other volume between the A and the B controller, and then alternating servers in the NSD stanza creation file, caused only one server to be utilized. Both are considered standard practice, but all volumes ended up being on one server when Spectrum Scale created the NSD volumes. The solution in this case is to alternate every two volumes between the A and B controller. Another solution is to adjust the stanza file, ensuring that they alternate correctly.

Figure 7) EF600 RDMA interface.



## Example testing methodology – CPOC EF600

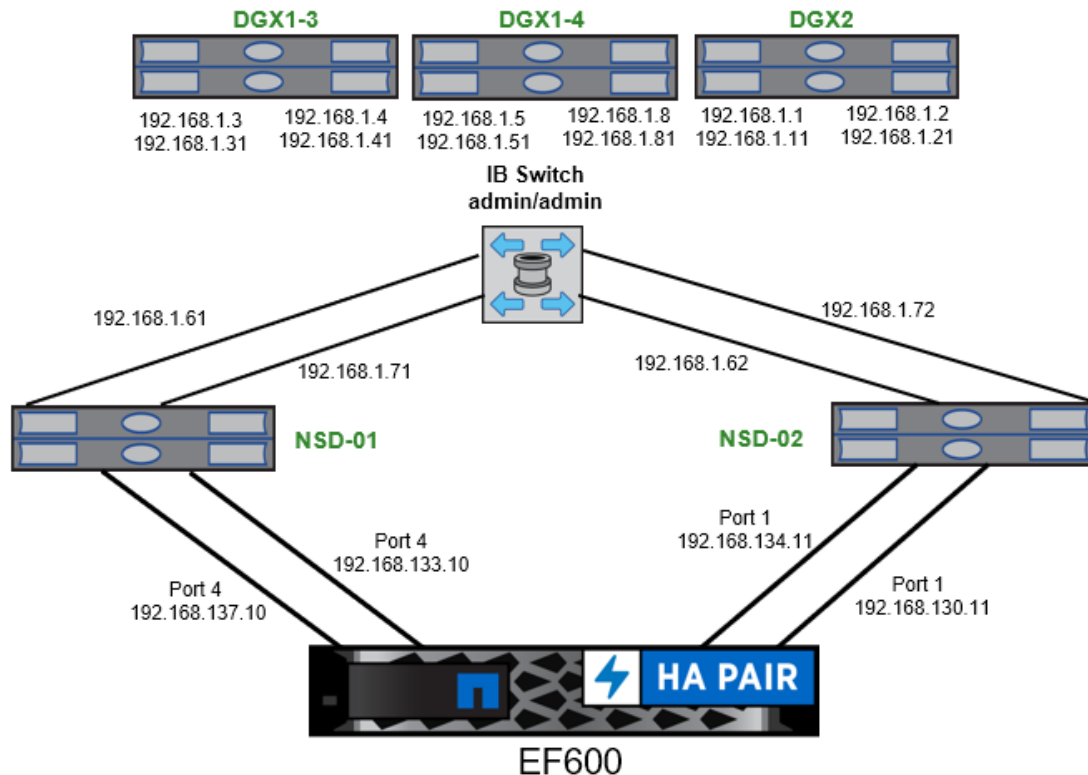
The example in this section uses three NVIDIA DGX servers as Spectrum Scale clients connected to two Fujitsu NSD servers via 100Gb IB with direct connections to the EF600 via two HCA ports.

### Configuration

Each client has multiple IB HCAs with four paths each going to an IB switch. The NSD servers will route two IB connections to the switch with two being directly connected to the EF600, as shown in Figure 8.



Figure 8) CPOC EF600 configuration.



## EF600 configuration

The EF600 is configured as follows:

- Two 10+2 RAID 6 volume groups – GPFS\_data\_vg1 and GPFS\_data\_vg2:
  - 24 1.9T NVMe drives
  - 512KiB segment size
  - Disable dynamic cache read prefetch
  - Secure-enabled = No
- Four 3567.00GiB volumes per volume group – 8 volumes total:
  - 20% free capacity left at the end of the volume group
- Create two hosts and assign under a top-level Cluster group from SANtricity:
  - Volumes are assigned under the Cluster group; both NSD servers see all volumes

## NSD configuration file

In the example setup, the NSD devices contain both metadata and data on all volumes. The NSD device primary server is alternated for HA and performance considerations.

```
root@dgx2-1:~# cat StanzeFileEF600-v5
%nsd: device=/dev/dm-0
nsd=VOLUME_001_gpfs_vla
servers=nsd-01-p1,nsd-02-p1
usage=dataAndMetadata
pool=system
failureGroup=1
```

```

%nsd: device=/dev/dm-3
nsd=VOLUME_001_gpfs_v1b
servers=nsd-02-p1,nsd-01-p1
usage=dataAndMetadata
pool=system
failureGroup=1

%nsd: device=/dev/dm-2
nsd=VOLUME_001_gpfs_v1c
servers=nsd-01-p1,nsd-02-p1
usage=dataAndMetadata
pool=system
failureGroup=1

%nsd: device=/dev/dm-6
nsd=VOLUME_001_gpfs_v1d
servers=nsd-02-p1,nsd-01-p1
usage=dataAndMetadata
pool=system
failureGroup=1

%nsd: device=/dev/dm-1
nsd=VOLUME_001_gpfs_v2a
servers=nsd-01-p1,nsd-02-p1
usage=dataAndMetadata
pool=system
failureGroup=1

%nsd: device=/dev/dm-5
nsd=VOLUME_001_gpfs_v2b
servers=nsd-02-p1,nsd-01-p1
usage=dataAndMetadata
pool=system
failureGroup=1

%nsd: device=/dev/dm-3
nsd=VOLUME_001_gpfs_v2c
servers=nsd-01-p1,nsd-02-p1
usage=dataAndMetadata
pool=system
failureGroup=1

%nsd: device=/dev/dm-7
nsd=VOLUME_001_gpfs_v2d
servers=nsd-02-p1,nsd-01-p1
usage=dataAndMetadata
pool=system
failureGroup=1

root@dgx2-1: #

```

## File system attributes

A 16MB block size was selected to test for large block I/O. Scatter was set for consistent performance throughout the file system, as shown in Table 5. Here is an example command creating a file system with a 16M block size and using the scatter allocation type:

```
mmcrfs cpocgpfs2 -F /usr/lpp/mmfs/EF600StanzaFile3 -B 16M -j scatter -T /gpfs2
```

**Table 5) Spectrum Scale file system configuration parameters.**

File System Attribute	Value
Block size	16777216
Block allocation type	scatter

## Validating infrastructure configuration with cache reads to the EF600

You must validate the infrastructure before running actual performance tests. One method of validation is to use `fio` with a very small block size, such that all data is held in the array's cache. Running a test set on the NSD servers will show the total throughput available from each server. You can run the same tests from the clients to show overall throughput capabilities from client to NSD.

To use `fio`, create a parameter file with the following global parameters, and then add jobs pointing to the file system. Using a size of 16m should keep everything in the array cache to validate the performance of the infrastructure, and adding multiple jobs will fill the pipeline.

```
root@dgx1-3:/home/fio-testing/fio-3.15# cat EF600cacheread.parm
[global]
rw=randread
size=16m
iodepth=16
direct=1
sync=0
runtime=120
bs=512K
loops=1000
numjobs=1

[GPFS_data_v1a]
filename=/gpfs2/dgx1-3a

[GPFS_data_v1b]
filename=/gpfs2/dgx1-3b

[GPFS_data_v1c]
filename=/gpfs2/dgx1-3c

[GPFS_data_v1d]
filename=/gpfs2/dgx1-3d

[GPFS_data_v2a]
filename=/gpfs2/dgx1-3e

[GPFS_data_v2b]
filename=/gpfs2/dgx1-3f

[GPFS_data_v2c]
filename=/gpfs2/dgx1-3g

[GPFS_data_v2d]
filename=/gpfs2/dgx1-3h

[GPFS_data_v3a]
filename=/gpfs2/dgx1-3i

[GPFS_data_v3b]
filename=/gpfs2/dgx1-3j

[GPFS_data_v3c]
filename=/gpfs2/dgx1-3k

[GPFS_data_v3d]
filename=/gpfs2/dgx1-3l

[GPFS_data_v4a]
filename=/gpfs2/dgx1-3m

[GPFS_data_v4b]
filename=/gpfs2/dgx1-3n

[GPFS_data_v4c]
filename=/gpfs2/dgx1-3o

[GPFS_data_v4d]
```

```

filename=/gpfs2/dgx1-3p

[GPFS_data_v5a]
filename=/gpfs2/dgx1-3q

[GPFS_data_v5b]
filename=/gpfs2/dgx1-3r

[GPFS_data_v5c]
filename=/gpfs2/dgx1-3s

[GPFS_data_v5d]
filename=/gpfs2/dgx1-3t

root@dgx1-3:/home/fio-testing/fio-3.15#

```

Run `fio` using the parameter file with 100Gb IB links. Up to 10GBs of bandwidth is possible. The following example shows 19GBps, so both links are active and up to full potential. Run the same test on each NSD to validate the infrastructure.

```

# ./fio EF600cacheread.parm
nsd-01:/home/mbostock/fio-testing/fio-3.15 # ./fio EF600cacheread.parm
GPFS_data_v1a: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v1b: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v1c: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v1d: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v2a: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v2b: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v2c: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v2d: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v3a: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v3b: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v3c: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v3d: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v4a: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v4b: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v4c: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v4d: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v5a: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v5b: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v5c: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
GPFS_data_v5d: (g=0): rw=randread, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB,
ioengine=psync, iodepth=16
fio-3.15
Starting 20 processes
...
Run status group 0 (all jobs):
  READ: bw=18.6GiB/s (19.0GB/s), 954MiB/s-1167MiB/s (1000MB/s-1223MB/s), io=313GiB (336GB),
run=13716-16779msec

```

## FIO test attributes

Ensure that `fio` is installed in the same location on each server in the cluster. You can create scripts to do simultaneous testing across multiple clients. Examples are in Appendix A. The following settings are typical for testing the configuration, with the number of jobs cycled through that increase by four per cycle. File size was set to 16GB, with a 16MB block size:

- Testing done with `fio` scripts utilizing the following attributes:
  - `njob_array=( 4 8 12 16 20 24 28 32 36 40 44 48 52)`
  - `file_size_per_job=$(( 16384 * (1024 * 1024) ))`
  - `blocksize=$(( 16 * (1024 * 1024) ))`
- NSD volumes alternating mapping to servers for high availability.

## Array performance

Using the attributes for `fio` defined in the section titled “File system attributes,” and sending the output to a file for parsing, you can create graphs to show the performance of the cluster.

The graphs in this section reflect the performance of the array during the test run. Figure 9 shows sequential writes, Figure 10 shows sequential reads, and Figure 11 shows random read results.

**Figure 9) Sequential writes.**

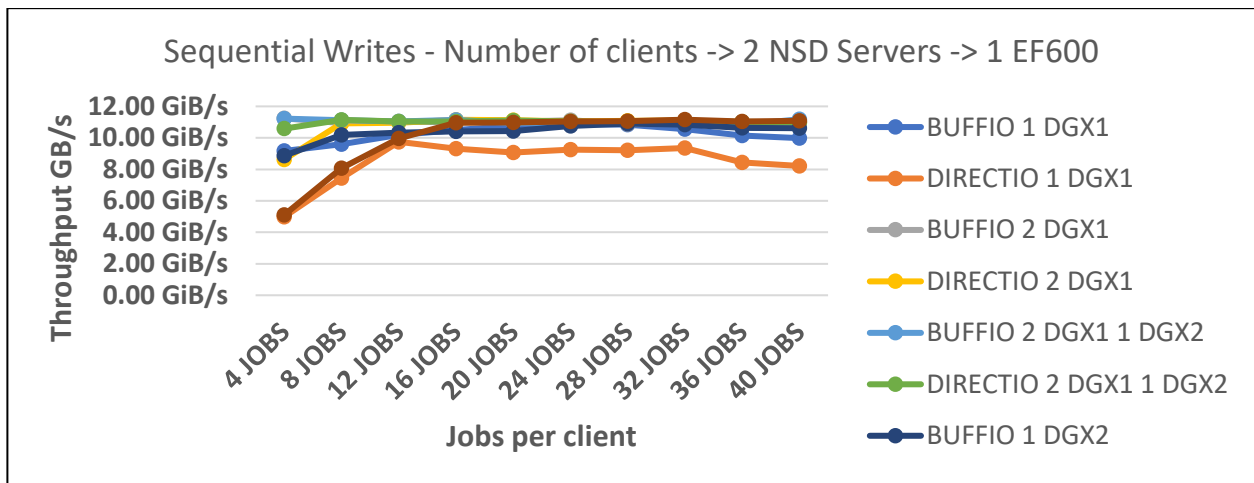


Figure 10) Sequential reads.

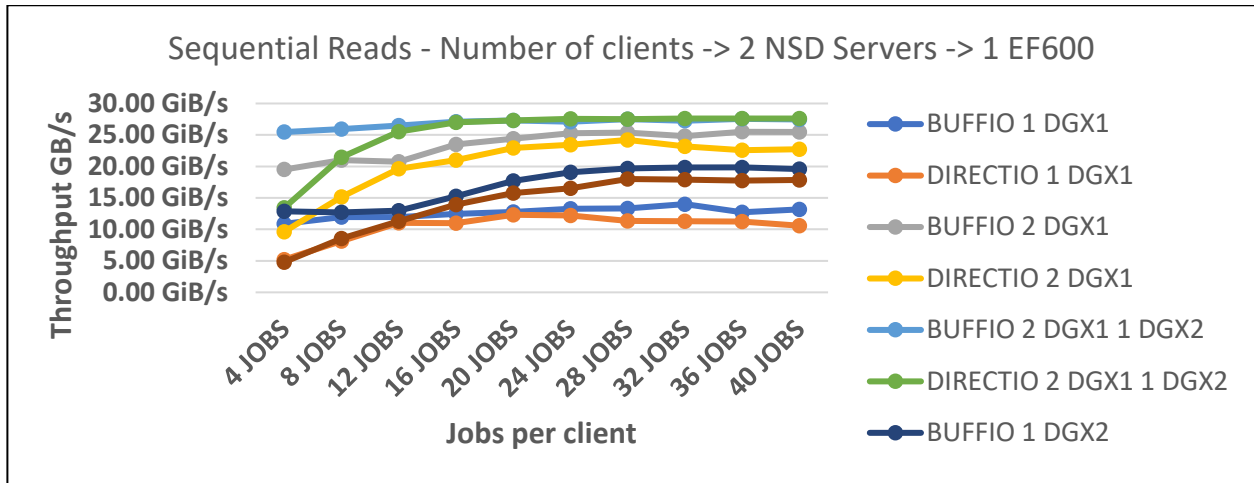
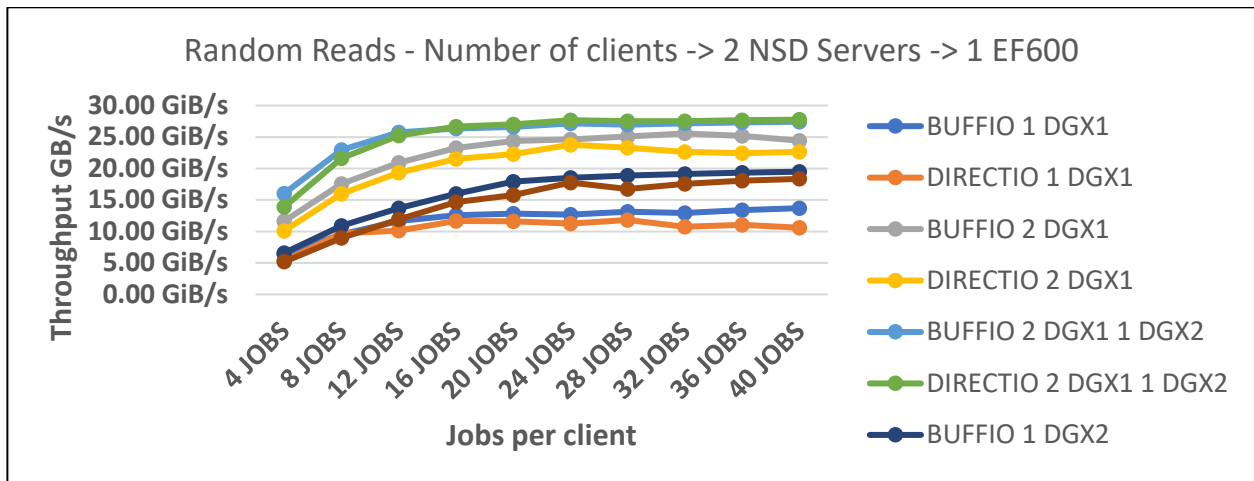


Figure 11) Random reads.



## Appendix A: FIO scripts

The following files are FIO scripts for running multi server performance tests creating output files and a parser to create a csv file output.



Multi\_client\_MAX\_PERF\_TEST.sh



pars\_fio\_multi-server.sh

## Appendix B: mmfslinux.ko error after upgrading OS

If you apply an operating system upgrade that changes the kernel, it might cause Spectrum Scale to fail at startup with a `mmfslinux.ko` failure. IBM provides a tool, `/usr/lpp/mmfs/bin/mmbuildgpl`, to rebuild the GPFS portability layer and allow startup to complete.

```
root@dgxl-3:~# mmstartup
Wed Mar  4 08:54:34 EST 2020: mmstartup: Starting GPFS ...
mmremote: startSubsys: The /lib/modules/4.15.0-88-generic/extra/mmfslinux.ko kernel extension
does not exist. Use mmbuildgpl command to create the needed kernel extension for your kernel or
copy the binaries from another node with the identical environment.
mmremote: startSubsys: Unable to verify kernel/module configuration.
mmstartup: Command failed. Examine previous error messages to determine cause.
root@dgxl-3:~# /usr/lpp/mmfs/bin/mmbuildgpl
-----
mmbuildgpl: Building GPL (5.0.3.1) module begins at Wed Mar  4 08:59:40 EST 2020.
-----
Verifying Kernel Header...
  kernel version = 41500088 (41500088000000, 4.15.0-88-generic, 4.15.0-88)
  module include dir = /lib/modules/4.15.0-88-generic/build/include
  module build dir   = /lib/modules/4.15.0-88-generic/build
  kernel source dir  = /usr/src/linux-4.15.0-88-generic/include
  Found valid kernel header file under /lib/modules/4.15.0-88-generic/build/include
Verifying Compiler...
  make is present at /usr/bin/make
  cpp is present at /usr/bin/cpp
  gcc is present at /usr/bin/gcc
  g++ is present at /usr/bin/g++
  ld is present at /usr/bin/ld
make World ...
make InstallImages ...
-----
mmbuildgpl: Building GPL module completed successfully at Wed Mar  4 08:59:56 EST 2020.
-----
root@dgxl-3:~#
root@dgxl-3:~# mmstartup
Wed Mar  4 09:01:07 EST 2020: mmstartup: Starting GPFS ...
root@dgxl-3:~# mmhealth cluster show
```

Component	Total	Failed	Degraded	Healthy	Other
NODE	5	0	0	4	1
GPFS	5	0	0	4	1
NETWORK	4	0	0	4	0
FILESYSTEM	1	0	0	1	0
DISK	4	0	0	4	0
PERFMON	4	0	0	4	0
THRESHOLD	4	0	0	4	0

## Appendix C: IB port state down

If an IB port will not go to a UP state using the `ifconfig` command, and the physical state shows up, you can try issuing an `ibportstate reset` to bring up the port:

```
[root@ictm0803h11-client1 ~]# ip a |grep ib
6: ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2044 qdisc mq state UP group default qlen 256
    link/infiniband 00:00:03:93:fe:80:00:00:00:00:00:00:01:98:03:9b:03:00:71:f7:68 brd
    00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
    inet 192.168.100.103/24 brd 192.168.100.255 scope global noprefixroute ib0
7: ib1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 4092 qdisc mq state DOWN group default qlen 256
    link/infiniband 00:00:0b:bd:fe:80:00:00:00:00:00:00:00:98:03:9b:03:00:71:f7:69 brd
    00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
    inet 192.168.200.103/24 brd 192.168.200.255 scope global noprefixroute ib1
[root@ictm0803h11-client1 ~]# ibstat
CA 'mlx5_0'
```

```

CA type: MT4121
Number of ports: 1
Firmware version: 16.26.1040
Hardware version: 0
Node GUID: 0x98039b030071f768
System image GUID: 0x98039b030071f768
Port 1:
    State: Active
    Physical state: LinkUp
    Rate: 100
    Base lid: 12
    LMC: 0
    SM lid: 17
    Capability mask: 0x2659e84a
    Port GUID: 0x98039b030071f768
    Link layer: InfiniBand
CA 'mlx5_1'
CA type: MT4121
Number of ports: 1
Firmware version: 16.26.1040
Hardware version: 0
Node GUID: 0x98039b030071f769
System image GUID: 0x98039b030071f768
Port 1:
    State: Active
    Physical state: LinkUp
    Rate: 100
    Base lid: 15
    LMC: 0
    SM lid: 17
    Capability mask: 0x2659e84a
    Port GUID: 0x98039b030071f769
    Link layer: InfiniBand
[root@ictm0803h11-client1 ~]# ibportstate -G 0x98039b030071f769 1 reset
Initial CA/RT PortInfo:
# Port info: Lid 15 port 1
LinkState:.....Active
PhysLinkState:.....LinkUp
Lid:.....15
SMLid:.....17
LMC:.....0
LinkWidthSupported:.....1X or 4X
LinkWidthEnabled:.....1X or 4X
LinkWidthActive:.....4X
LinkSpeedSupported:.....2.5 Gbps or 5.0 Gbps or 10.0 Gbps
LinkSpeedEnabled:.....2.5 Gbps or 5.0 Gbps or 10.0 Gbps
LinkSpeedActive:.....10.0 Gbps
LinkSpeedExtSupported:.....14.0625 Gbps or 25.78125 Gbps
LinkSpeedExtEnabled:.....14.0625 Gbps or 25.78125 Gbps
LinkSpeedExtActive:.....25.78125 Gbps
Mkey:.....<not displayed>
MkeyLeasePeriod:.....0
ProtectBits:.....0
# MLNX ext Port info: Lid 15 port 1
StateChangeEnable:.....0x00
LinkSpeedSupported:.....0x01
LinkSpeedEnabled:.....0x01
LinkSpeedActive:.....0x00

After PortInfo set:
# Port info: Lid 15 port 1
LinkState:.....Down
PhysLinkState:.....Polling
Lid:.....15
SMLid:.....17
LMC:.....0
LinkWidthSupported:.....1X or 4X
LinkWidthEnabled:.....1X or 4X
LinkWidthActive:.....4X
LinkSpeedSupported:.....2.5 Gbps or 5.0 Gbps or 10.0 Gbps
LinkSpeedEnabled:.....2.5 Gbps or 5.0 Gbps or 10.0 Gbps

```



```

LinkSpeedActive:.....Extended speed
LinkSpeedExtSupported:.....14.0625 Gbps or 25.78125 Gbps
LinkSpeedExtEnabled:.....14.0625 Gbps or 25.78125 Gbps
LinkSpeedExtActive:.....25.78125 Gbps
Mkey:.....<not displayed>
MkeyLeasePeriod:.....0
ProtectBits:.....0
[root@ictm0803h11-client1 ~]# ip a |grep ib
6: ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2044 qdisc mq state UP group default qlen 256
    link/infiniband 00:00:03:93:fe:80:00:00:00:00:00:01:98:03:9b:03:00:71:f7:68 brd
    00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
    inet 192.168.100.103/24 brd 192.168.100.255 scope global noprefixroute ib0
7: ib1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2044 qdisc mq state UP group default qlen 256
    link/infiniband 00:00:0b:bd:fe:80:00:00:00:00:00:00:01:98:03:9b:03:00:71:f7:69 brd
    00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
    inet 192.168.200.103/24 brd 192.168.200.255 scope global noprefixroute ib1
[root@ictm0803h11-client1 ~]#

```

If the network does not show as up, try running the `ifconfig ib1 up` command and recheck it. If it still does not come up, there might be a hardware or connectivity issue.

## Appendix D: SuperPod cluster settings

The following table shows some additional settings that were used in testing the SuperPod configuration:

**Table 6) SuperPod cluster attributes.**

Cluster Attribute	Value	Cluster Attribute	Value
dmapiFileHandleSize	32	nsdThreadsPerDisk	24
ccrEnabled	yes	nsdSmallThreadRatio	1
verbsRdma	enable	nsdThreadsPerQueue	128
maxblocksize	16M	numaMemoryInterleave	yes
enforceFilesetQuotaOnRoot	no	maxMBpS	100000
workerThreads	512	maxFilesToCache	200000
verbsRdmasPerNode	1024	nsdbufspace	70
verbsRdmasPerConnection	16	maxBufferDescs	32k
prefetchThreads	1024	maxStatCache	5000

## Appendix E: Setting up tiering

Spectrum Scale supports tiering with an installable feature called Transparent Cloud Tiering (TCT). TCT allows for automated data management through the Spectrum Scale Information Lifecycle Management (ILM) policy scan engine. You can set policies that will migrate your data from a hot tier to a cold tier. In the use case described in this appendix, a Simple Storage Service (S3) container was the cold tier and a NetApp® EF570 all-flash array was the hot tier.

### Prerequisites

To set up tiering, use one of the Spectrum Scale editions that support cloud services, such as the Advanced Edition, Data Management Edition, Developer Edition, or Erasure Code Edition. These are the only editions that come with the required cloud services `.rpm` package (Red-Hat Package Manager).

1. Install the `gpfs.tct.server.rpm` package (which is the cloud services `.rpm` package you need) on the servers to be used as cloud service nodes. The `.rpm` package can be found in the `gpfs_rpms` folder of your installer directory. This is usually under `/usr/lpp/mmfs/`.

```
[root@ictml619h4 Desktop]# rpm -ivh gpfs.tct.server-1.1.8.1.el7.x86_64.rpm
Preparing...                               ##### [100%]
Updating / installing...
 1:gpfs.tct.server-1.1.8.1                ##### [100%]
RPM installation logs are at /var/log/mcstore/gpfs.tct.server-rpm.log.
[root@ictml619h4 Desktop]#
```

2. Then you install the `gpfs.tct.client` .rpm package on all the remaining servers in the cluster. The file is in the `gpfs_rpms` folder of your installer directory.

```
[root@ictml618h15 Desktop]# rpm -ivh gpfs.tct.client-1.1.8.1.el7.x86_64.rpm
Preparing...                               ##### [100%]
Updating / installing...
 1:gpfs.tct.client-1.1.8.1                ##### [100%]
[root@ictml618h15 Desktop]#
```

## Designate a TCT node class

When you designate a TCT node class, you specify the nodes that handle data transfer between the Spectrum Scale cluster and the cloud. The node class can have a maximum of four nodes per Spectrum Scale cluster. These nodes can be a combination of Cluster Extension Services (CES) and Network Storage Device (NSD) nodes.

To create the node class, you run the `mmcrnodeclass` command. You can verify that the node class has been created by running `mmclsnodeclass`.

```
ictml619h19:/usr/lpp/mmfs/bin # ./mmcrnodeclass node_class_1 -N ictml619h4-p0
mmcrnodeclass: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
ictml619h19:/usr/lpp/mmfs/bin # ./mmclsnodeclass
Node Class Name      Members
-----
node_class_1         ictml619h4-p0
```

## Designate cloud service nodes

After you define a node class, you need to designate all the nodes in that class as cloud service nodes.

1. You run the `mmchnode` command with the `--cloud-gateway-enable` flag to designate all the nodes in the TCT node class as cloud service nodes.
2. A cloud service node must have a GPFS server license. You enable this license by issuing the `mmchlicense` command.

```
ictml619h19:/usr/lpp/mmfs/bin # ./mmchlicense

Summary information
-----
Number of nodes defined in the cluster:          4
Number of nodes with server license designation: 3
Number of nodes with FPO license designation:    0
Number of nodes with client license designation: 1
Number of nodes still requiring server license designation: 0
Number of nodes still requiring client license designation: 0
This node runs IBM Spectrum Scale Data Management Edition
ictml619h19:/usr/lpp/mmfs/bin # ./mmchlicense server --accept -N ictml619h4-p0

The following nodes will be designated as possessing server licenses:
  ictml619h4-p0
mmchlicense: Command successfully completed
mmchlicense: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
ictml619h19:/usr/lpp/mmfs/bin # ./mmchnode -N node_class_1 --cloud-gateway-enable
Fri Feb 12 08:15:45 CST 2021: mmchnode: Processing node ictml619h4-p0
mmchnode: Verifying media for Transparent Cloud Tiering nodes...
mmchnode: node ictml619h4-p0 media checks passed.
```

```
mmchnode: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
ictml619h19:/usr/lpp/mmfs/bin #
```

## Start cloud services

Start cloud services by running `mmcloudgateway`.

```
ictml619h19:/usr/lpp/mmfs/bin # ./mmcloudgateway service start -N node_class_1
mmcloudgateway: Migrating configuration data to the new format. This is a one time operation.
Cloud node class 'node_class_1' did not have configuration data to migrate
mmcloudgateway: Configuration data migration complete.
mmcloudgateway: Sending the command to node ictml619h4-p0.
Starting the Transparent Cloud Tiering service...
mmcloudgateway: The command completed on node ictml619h4-p0.

mmcloudgateway: Command completed.
ictml619h19:/usr/lpp/mmfs/bin #
```

## Define a cloud storage access point

A cloud storage access point (CSAP) associates your cloud account with your object storage.

1. To create a cloud account, use the `mmcloudgateway` command, as shown in this example:

```
ictml619h19:/usr/lpp/mmfs/bin # ./mmcloudgateway account create --cloud-nodeclass node_class_1 --
account-name cloud_account_1 --cloud-type S3 --username Admin --pwd-file pwd.txt
mmcloudgateway: Sending the command to the first successful node starting with ictml619h4-p0
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on ictml619h4-p0.
mmcloudgateway: You can now delete the password file 'pwd.txt'
mmcloudgateway: Command completed.
ictml619h19:/usr/lpp/mmfs/bin #
```

2. To create a CSAP, use the `mmcloudgateway` command as shown in this example:

```
ictml619h19:/usr/lpp/mmfs/bin # ./mmcloudgateway cloudStorageAccessPoint create --cloud-nodeclass
node_class_1 --cloud-storage-access-point-name minio_S3 --account-name cloud_account_1 --url
http://10.113.72.104:9000
mmcloudgateway: Sending the command to the first successful node starting with ictml619h4-p0
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on ictml619h4-p0.
mmcloudgateway: Command completed.
ictml619h19:/usr/lpp/mmfs/bin #
```

## Create a cloud service

By creating a cloud service, users are able to issue Spectrum Scale commands to move data. You must have at least one cloud service for each cloud account.

```
ictml619h19:/usr/lpp/mmfs/bin # ./mmcloudgateway cloudService create --cloud-nodeclass
node_class_1 --cloud-service-name minio_cloud --cloud-service-type Tiering --account-name
cloud_account_1
mmcloudgateway: Sending the command to the first successful node starting with ictml619h4-p0
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on ictml619h4-p0.
mmcloudgateway: Command completed.
ictml619h19:/usr/lpp/mmfs/bin #
```

## Create a container pair set

To bind your cloud services to the data on the cluster, you create a container pair set on your cloud storage. One of the two containers is for data and the other is for metadata. These containers cannot be shared with other file systems or clusters.

```
ictml619h19:/usr/lpp/mmfs/bin # ./mmcloudgateway containerpairset create --cloud-nodeclass
node_class_1 --container-pair-set-name new_container --cloud-service-name minio_cloud --scope-to-
filesystem --data-container datacontainer --meta-container metacontainer --path /gpfs0/
mmcloudgateway: Sending the command to the first successful node starting with ictml619h4-p0
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on ictml619h4-p0.
mmcloudgateway: Command completed.
ictml619h19:/usr/lpp/mmfs/bin #
```

## Create an ILM policy

You use an ILM policy to automate data management. A policy can regulate the execution of actions such as initial file placement, file migration, and file deletions. You create policy rules by using SQL-like statements that use file attributes such as size, access time, and name as criteria for executing policies.

You create a policy file with the rules and criteria needed for your automated file management. Sample policies that can be used for reference can be found at `/opt/ibm/MCstore/samples`. The following example of creating an ILM policy has several steps, starting with configuring some settings. Then it specifies rules for how the `THRESHOLD` keyword works in the policy.

1. Before dealing directly with the threshold, enable `lowDiskSpace` events.

```
ictml619h19:/usr/lpp/mmfs/bin # ./mmchconfig enableLowSpaceEvents=yes
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
ictml619h19:/usr/lpp/mmfs/bin #
```

2. Restart Spectrum Scale by issuing `mmshutdown -a` followed by `mmstartup -a`.
3. Add a callback (to trigger the policy system when the low space event is generated) by using the `mmaddcallback` command.

```
ictml619h19:/usr/lpp/mmfs/bin # ./mmaddcallback MIGRATION --command
/usr/lpp/mmfs/bin/mmstartpolicy --event lowDiskSpace --parms "%eventName %fsName --single-
instance"
mmaddcallback: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
ictml619h19:/usr/lpp/mmfs/bin #
```

4. The following sample policy allows migration of files that are bigger than 40KB and that have not been accessed in over 30 days. The threshold is set up so that when the pool is 80% full, it will start migrating data and stop when the pool is 60% full.

```
/* Sample policy file for threshold migration*/

/*List files you don't want to be migrated*/
define(
  exclude_list,
  (
    FALSE
    OR PATH_NAME LIKE '%/.%/%'
  )
)

/*This rule defines the external pool for migration.*/
RULE EXTERNAL POOL 'minio_cloud' EXEC '/opt/ibm/MCStore/bin/mcstore' OPTS '-F'

/*Migration will start when the pool "testpool1" is 80% full and will continue migration
until the pool is 60% full. The files that will be migrated are bigger than 40KB and have not
been
accessed in over 30 days*/

RULE 'MoveLargeColdData' MIGRATE FROM POOL 'testpool1'
THRESHOLD(80,60)
TO POOL 'minio_cloud'
WHERE (KB_ALLOCATED>40)
AND NOT (exclude_list)
```

```
AND (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '30' DAYS)
/*Default placement of file if no other placement rules apply*/
RULE 'Placement' SET POOL 'testpool1'
```

#### 5. To validate the policy, run mmchpolicy.

```
ictml619h19:/usr/lpp/mmfs/bin # ./mmchpolicy gpfs0 ../policy.txt -I test
Validated policy 'policy.txt': Parsed 3 policy rules.
ictml619h19:/usr/lpp/mmfs/bin #
```

#### 6. Install the policy.

```
ictml619h19:/usr/lpp/mmfs/bin # ./mmchpolicy gpfs0 ../policy.txt -I yes
Validated policy 'policy.txt': Parsed 3 policy rules.
Policy 'policy.txt' installed and broadcast to all nodes.
ictml619h19:/usr/lpp/mmfs/bin #
```

## Where to find additional information

To learn more about the information that is described in this document, review the following documents and/or websites:

- E-Series and SANtricity 11 Documentation Center  
<https://docs.netapp.com/ess-11/index.jsp>
- E-Series and SANtricity documentation resources  
<https://www.netapp.com/documentation/eseries-santricity/>
- NetApp Product Documentation  
<https://docs.netapp.com>

## Version history

Version	Date	Document version history
Version 1.0	September 2020	Initial release.
Version 1.1	January 2021	Changed status from internal to external. Reformatted the document.
Version 1.2	March 2021	Added Appendix E: Setting up tiering.

Refer to the [Interoperability Matrix Tool \(IMT\)](#) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

### **Copyright Information**

Copyright © 2020–2021 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

Data contained herein pertains to a commercial item (as defined in FAR 2.101) and is proprietary to NetApp, Inc. The U.S. Government has a non-exclusive, non-transferrable, non-sublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b).

### **Trademark Information**

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.

TR-4859-0321